Wijmo 5

version 5.20173.380

Table of Contents

Welcome to Wijmo

Wijmo represents a new generation of JavaScript controls. It takes full advantage of the latest HTML5 technologies, making no compromises to support legacy browsers. The result is a set of controls that are much faster, smaller, and easier to use than what was possible before.

Wijmo has no dependencies other than EcmaScript5. You can use it without jQuery, jQueryUI, or any other frameworks.

Wijmo requires modern browsers (IE9 or better) to leverage the following technologies:

- **ECMAScript 5:** The ECMAScript 5 standard adds support for property getters and setters. This may seem like a small change, but it makes a huge difference. For example, instead of writing `control.value(control.value() + 1)` to increment the value of a property, now you can write `control.value++`. The ECMAScript 5 standard adds many other significant enhancements, like the bind (http://javascriptissexy.com/javascript-apply-call-and-bind-methods-are-essential-for-javascript-professionals/) method that allows you to specify the value of the 'this' parameter in your callbacks. There are also new array methods (https://www.inkling.com/read/javascript-definitive-guide-david-flanagan-6th/chapter-7/ecmascript-5-array-methods) that can save a lot of time.
- **SVG:** Modern browsers implement **SVG**, which makes it easier to create amazing visual representations of your data. Wijmo leverages SVG directly, without the overhead that would be required if it had to support legacy browsers.
- **TypeScript:** We wrote Wijmo in **TypeScript**, taking advantage of type-checking and OOP concepts such as modules, classes, and inheritance. The output is still pure JavaScript, so you can use either language in your own development work.
- **Mobile Devices:** Wijmo was designed with mobile browser support built in from the start. Responsive layouts and touch support were major considerations in the design and implementation of every Wijmo control.
- **AngularJS: AngularJS** is one of the most popular and powerful JavaScript application frameworks today. We have supported Angular 1.x since it was released, and Angular 2 even before it was released! For more information, see the Using Wijmo in AngularJS Applications topic.
- **Other frameworks:** You can use Wijmo with any other JavaScript frameworks you like. In addition to AngularJS 1.x and 2.x, we ship interop modules for **React**, **Vue**, and **KnockoutJS**. We plan to add other frameworks to this list in the future, based on customer requests.
- **Bootstrap: Bootstrap** is one of the easiest, most powerful, and most popular CSS frameworks available. We use it in our samples and in our on-line documentation. If you use Bootstrap, be assured that Wijmo will blend right in with no extra effort required on your part.

We do realize that some scenarios require support for legacy browsers (IE8 and earlier). For that reason, we will continue to maintain versions of the original Wijmo for as long as our customers require it. If you need to support IE8 and earlier, keep using the original Wijmo. We will maintain and support it as usual. If you are ready to move to HTML5 and modern browsers, Wijmo is for you!

This site contains detailed information about Wijmo's APIs. If you are looking for conceptual information and practical examples, please check our **LearnWijmo application**.

Referencing Wijmo in Your Applications

To use Wijmo in your applications, include it by adding a few references to your HTML pages.

The minimal set of files required by any Wijmo application is:

- **wijmo.js:** Contains the Wijmo infrastructure including the Globalize, Event, Control, and CollectionView classes.
- **wijmo.css:** Contains the CSS rules used for styling all Wijmo controls.

In addition to these, include one or more additional files, depending on which components you use:

- **wijmo.grid.js:** Contains the FlexGrid control.
- **wijmo.chart.js:** Contains the FlexChart and FlexPie controls.
- **wijmo.input.js:** Contains several input controls, including ComboBox, AutoComplete, InputDate, InputTime, InputNumber, InputMask, ListBox, Menu, and Calendar controls.
- **wijmo.gauge.js:** Contains several gauge controls, including LinearGauge, RadialGauge, and BulletGraph.
- **angular.js:** Google's AngularJS framework, required for AngularJS applications.
- **wijmo.angular.js:** Contains AngularJS directives that allow you to use Wijmo controls directly in your markup.
- **wijmo.culture.*[CultureName]*.js:** Contains culture-specific files used to develop applications in languages other than American English.
- **wijmo.theme.*[ThemeName]*.css:** Contains CSS rules used to customize the appearance of the Wijmo controls.

As for the actual location of the files, you have two options. You may download Wijmo and copy the required files to the appropriate folders within your application, or you may reference Wijmo files hosted in the cloud, on our Content Delivery Network (CDN). The sections below show examples.

Deploying Wijmo locally

Download the Wijmo files and copy them to a folder within your application. If you place the Wijmo script files in a folder called "scripts/vendors," and the css files in a folder called "styles," you can add the following lines to your HTML pages:

```
<!-- Wijmo references (required) -->
<script src="scripts/vendor/controls/wijmo.min.js"></script>
<link href="styles/wijmo.min.css" rel="stylesheet"/>

<!-- Wijmo controls (optional, include the controls you need) -->
<script src="scripts/vendor/controls/wijmo.grid.min.js"></script>
<script src="scripts/vendor/controls/wijmo.chart.min.js"></script>
<script src="scripts/vendor/controls/wijmo.input.min.js"></script>
<script src="scripts/vendor/controls/wijmo.gauge.min.js"></script>

<!-- Wijmo custom theme (optional, include the theme you like) -->
<link href="styles/themes/wijmo.theme.modern.min.css" rel="stylesheet"/>

<!-- Wijmo custom culture (optional, include the culture you want) -->
<script src="scripts/vendor/controls/cultures/wijmo.culture.ja.min.js"></script>

<!-- AngularJS and Wijmo directives (optional, use in AngularJS applications) -->
<script src="scripts/vendor/angular.min.js"></script>
<script src="scripts/vendor/interop/angular/wijmo.angular.min.js"></script>
```

Deploying Wijmo from CDN

In this case, there is nothing to download. Simply add the following lines to your HTML pages:

```
<!-- Wijmo references (required) -->
<script src="http://cdn.wijmo.com/5.20142.15/controls/wijmo.min.js"></script>
<link href="http://cdn.wijmo.com/5.20142.15/styles/wijmo.min.css" rel="stylesheet"/>

<!-- Wijmo controls (optional, include the controls you need) -->
<script src="http://cdn.wijmo.com/5.20142.15/controls/wijmo.grid.min.js"></script>
<script src="http://cdn.wijmo.com/5.20142.15/controls/wijmo.chart.min.js"></script>
<script src="http://cdn.wijmo.com/5.20142.15/controls/wijmo.input.min.js"></script>
<script src="http://cdn.wijmo.com/5.20142.15/controls/wijmo.gauge.min.js"></script>

<!-- Wijmo custom theme (optional, include the theme you like) -->
<link href="http://cdn.wijmo.com/5.20142.15/styles/themes/wijmo.theme.modern.min.css" rel="stylesheet"/>

<!-- Wijmo custom culture (optional, include the culture you want) -->
<script src="http://cdn.wijmo.com/5.20142.15/controls/cultures/wijmo.culture.ja.min.js"></script>

<!-- AngularJS and Wijmo directives (optional, use in AngularJS applications) -->
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>
<script src="http://cdn.wijmo.com/5.20142.15/interop/angular/wijmo.angular.min.js"></script>
```

The CDN version includes a Wijmo watermark element at the bottom right of the screen. If you don't want to display the watermark, then you must deploy Wijmo locally as described above.

Note: The order of the references is important. The **wijmo.js** module must be the first, followed by the control modules, control extensions, and the **wijmo.angular** module should be included last.

Creating Wijmo Controls

Every Wijmo control is associated with an HTML element that hosts it on the page. To create a control, you start by adding a **div** element to the page, then use JavaScript code to instantiate the control and bind it to the host element.

For example, this fiddle shows how you can create a **FlexGrid** and a **FlexChart** and bind them to a small data source:

Example: Creating Controls

Show me (http://jsfiddle.net/Wijmo5/MWue8)

The fiddle includes all the necessary references (as described in the Referencing Wijmo in your Applications topic). The HTML part of the fiddle declares two **div** elements named 'theGrid' and 'theChart':

```
<h1>Hello</h1>

<p>This is a FlexGrid control:</p>
<div id="theGrid"></div>

<p>And this is a FlexChart:</p>
<div id="theChart"></div>

<p>That's it for now...</p>
```

The JavaScript part of the fiddle executes when the document has loaded. It creates a small data set, binds the controls to the **div** elements, then binds the controls to the data set:

```
<script id="scriptInit">
onload = function () {

  // generate some random data
  var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(','),
      data = [];
  for (var i = 0; i < countries.length; i++) {
    data.push({
      country: countries[i],
      downloads: Math.round(Math.random() * 20000),
      sales: Math.random() * 10000,
      expenses: Math.random() * 5000
    });
  }

  // create grid and show data
  var grid = new wijmo.grid.FlexGrid('#theGrid', {
    itemsSource: data
  });

  // create a chart and show the same data
  var chart = new wijmo.chart.FlexChart('#theChart', {
      itemsSource: data,
      bindingX: 'country',
      series: [
          { name: 'Sales', binding: 'sales' },
          { name: 'Expenses', binding: 'expenses' },
          { name: 'Downloads', binding: 'downloads', chartType: wijmo.chart.ChartType.LineSymbols } ]
  });
}
</script>
```

Notice that the size and position of the control are determined by the host element. In this case, we use CSS to set the grid's height to "auto," causing it to automatically size itself to its contents. We also set the **max-height** value so if there are too many items to fit the space the grid will automatically show scrollbars.

In most cases, you use a CSS framework such as Bootstrap to lay out your pages, and you lay out the controls exactly like any other HTML elements.

You can get a reference to the element that hosts a Wijmo control using the control's **hostElement** property. You can get a reference to the control being hosted by a given element using the **Control.getControl(element)** static method.

For more details on control sizing and layout, see the Sizing and Styling Controls topic.

You can use **div** elements as hosts for all Wijmo controls. Additionally, you can use **input** elements as hosts for the most input controls, and **select** elements as hosts for the **ListBox**, **ComboBox**, **AutoComplete**, and **Menu** controls.

Sizing and Styling Controls

Wijmo controls rely on CSS for styling, appearance, and sizing.

Because of this, Wijmo controls don't have properties such as "width," "height," or "background." Styling and layout is handled using CSS. If you are used to .NET controls, including **WinForms** and **XAML**, this may feel a little strange at first. But once you get used to CSS, you will find that it is very easy and extremely powerful. You can easily style all instances of a control type, or style a specific control, all with a few lines of extremely re-usable CSS.

Sizing Controls

The size and position of the controls are determined by the hosting element, which follows the usual HTML/CSS rules. For example, the CSS rule below stipulates that elements with class "grid" should have their height calculated automatically to fit the grid content, up to a limit of 300 pixels:

```
.grid {
    height: auto;
    max-height: 300px;
}
```

The fiddle below shows how this rule affects two **FlexGrid** controls. The first grid has only a few items, so it is resized to fit its content (like a regular HTML table). The second grid has a lot more items, so its height is automatically set to 300 pixels and it becomes scrollable.

Example: Sizing Controls

Show me (http://jsfiddle.net/Wijmo5/J4zME)

The first grid has only five elements to show. Since that requires less than 300 pixels, the grid shows all elements and doesn't need scrollbars. The second grid contains 10,000 items. That exceeds 300 pixels, so the grid becomes scrollable.

Styling Controls

Control styling follows the same logic as sizing. Use CSS to override fonts, colors, margins, padding, and pretty much any visual aspect of any part of the controls.

For example, this fiddle shows how you can modify the appearance of the **FlexGrid** control using CSS:

Example: Styling Controls

Show me (http://jsfiddle.net/Wijmo5/3L8Cf)

Notice how now the grids now have a plain black and white look. To do this, we change the CSS and specify the styles for elements **within** the grid. Wijmo controls assign class names to their constituent elements, which enables easy and flexible styling.

For example, the CSS below creates cell elements (elements with class "wj-cell" contained in elements with class "grid") with no border and a white background:

```
.grid .wj-cell {
    border: none;
    background-color: #fff;
}
```

Code-based Styling

Although the Wijmo controls rely on CSS for layout and sizing, there are a few situations where you may want to use code to get total control of some aspects of a control.

For example, the FlexGrid calculates the row heights based on the font being used to render the control. But you may want to override that CSS-based setting and specify the exact row heights yourself. You can do this by setting the following properties:

```
// set the height of rows in the scrollable area
flex.rows.defaultSize = 34;
// set the height of rows in the column header area
flex.columnHeaders.rows.defaultSize = 40;
```

This is shown in the fiddle below, which also uses CSS to achieve a very specific, customized look for the grid. The fiddle uses the "initialized" event to get a reference to the FlexGrid control. This is a convenient way to get a reference to the control when it is created with markup.

Example: Code-based Styling

Show me (http://jsfiddle.net/Wijmo5/kzrutyj9)

Pseudo Classes

CSS pseudo-classes are keywords added to selectors that specifies a special state of the element to be selected. For example, `:hover` will apply a style when the user hovers over the element specified by the selector.

Pseudo-classes are important in forms because they let you apply styles to elements not only in relation to the content of the document tree, but also in relation to external factors like whether the element has the focus (`:focus`) or is in an invalid state (`:invalid`).

Some of the standard pseudo-classes are limited in their usefulness because they apply only to specific elements, and not to the elements ancestors. For example, many Wijmo input controls contain input elements; when the input elements have the focus, the inner input element gets the `:focus` pseudo-class, but the host element that contains the control does not.

For this reason, Wijmo adds some pseudo-classes of its own to make building effective forms easier:

- **wj-state-focused**: Added to control host elements when the control **contains** the active element (not necessarily when the host element **is** the active element).
- **wj-state-invalid**: Added to control host elements when the control contains input elements in an invalid state.
- **wj-state-empty**: Added to control host elements when the control contains an input element with no content (this is different from the `:empty` pseudo-class which is applied to elements that have no children.
- **wj-state-readonly**: Added to control host elements when the control's **isReadOnly** property is set to true.
- **wj-state-disabled**: Added to control host elements when the control's **isDisabled** property is set to true (which corresponds to adding a "disabled" attribute to the control's host element).

The fiddle below shows an example of how you can use the **wj-state-focused** pseudo-class to apply CSS animations to the thumb element of linear and radial gauges when they get the focus.

Example: Pseudo Classes

Show me (http://jsfiddle.net/Wijmo5/450s0Lym)

Wijmo and HTML Events

HTML5 has an eventing mechanism that works for HTML elements, but cannot be used to add events to arbitrary objects, such as controls and collections.

Because of this, Wijmo defines an **Event** class that is used for implementing all events for all Wijmo classes. The main differences between Wijmo and HTML events are:

1. Wijmo events may be declared by any class (not just HTML elements).
2. Wijmo events are lighter than HTML events, because do not have routing (capturing and bubbling). They target only the object that declared the event.
3. You can add and remove Wijmo event handlers by calling the event's **addHandler** and **removeHandler** methods (as opposed to the **addEventListener** and **removeEventListner** methods used with HTML events).
4. Every Wijmo event handler takes two parameters: (a) the event sender, and (b) the event arguments.
5. Wijmo follows a pattern where event "XYZ" is raised by a corresponding method "onXYZ", which can be overridden by derived classes to handle the event without attaching any handlers or to customize or even suppress the event.

Wijmo events do not replace HTML events. Applications typically use both. HTML events are used to handle mouse and keyboard interactions that target a control's **hostElement** or elements defined in the control template. Wijmo events are used to handle control-specific events that are not directly related to the DOM. For example, **valueChanged** or **rowAdded**.

The example below shows how you can add handlers to HTML and Wijmo events on an **InputNumber** control using plain JavaScript:

```
// create the control
var ctl = new wijmo.input.InputNumber('#inputNumber');

// attach a Wijmo event handler
ctl.valueChanged.addHandler(function (s, e) {
    console.log('the value has changed to ' + s.value);
});

// attach an HTML event handler
ctl.addEventListener(ctl.hostElement, 'keypress', function(e) {
    console.log('you pressed ' + e.charCode);
});
```

The example above shows the syntax using plain JavaScript. Applications that use frameworks such as Angular, Knockout, Aurelia, or Vue have to use the syntax dictated by the framework.

For example, an Angular 1.x would attach a handler to the **valueChanged** Wijmo event this way:

```
<wj-input-number
    value-changed="myValueChangedEventHander(s, e)">...
```

In Angular2, you would do this instead:

```
<wj-input-number #theControl
    (value-changed)="myValueChangedEventHander(theControl, $event)">...
```

To find out more about HTML and Wijmo events, please see our **HTML and Wijmo Events** blog and the documentation for the Event class.

Using Enumerations in Wijmo

Several Wijmo controls have properties that take enumeration values.

For example, the **FlexChart**'s **chartType** property takes **wijmo.chart.ChartType** values.

Setting enumeration properties

The recommended way to set enumeration properties is as follows:

```
// setting the value of an enumeration property
chart.chartType = wijmo.chart.ChartType.Line;
```

The following alternatives are also valid and produce the same result:

```
// wijmo.chart.ChartType.Line has value 3:
chart.chartType = 3;

// enumerations are automatically parsed
chart.chartType = 'Line';
```

Getting enumeration properties

Getting the property will return 3 in all cases. If you want to get the value as a string (to show in the UI for example), you can do it as follows:

```
// getting the enumerated value as a number
console.log(chart.chartType); // outputs "3"

// getting the enumerated value as a string
console.log(wijmo.chart.ChartType[chart.chartType]); // outputs "Line"
```

Converting enumeration values

You can use the enumeration classes to convert between strings and the corresponding numbers by indexing. For example:

```
// convert enumeration value to string
console.log(wijmo.chart.ChartType[3]); // outputs "Line"
console.log(wijmo.chart.ChartType[1000]); // outputs "null"

// convert string to enumeration value
console.log(wijmo.chart.ChartType['Line']); // outputs "3"
console.log(wijmo.chart.ChartType['NoSuchValue']); // outputs "null"
```

Note for .NET Developers

The .NET, **Enum** class provides methods called **GetNames** and **GetValues** that return the names and values defined by any enumeration.

The code below shows how you could implement similar methods to get the names and values defined by TypeScript enumerations (as used in Wijmo):

```
// get the names defined by an enumeration
function getEnumNames(enumClass) {
    var names = [];
    for (var key in enumClass) {
        var val = parseInt(key);
        if (isNaN(val)) names.push(key);
    }
    return names;
}

// get the values defined by an enumeration
function getEnumValues(enumClass) {
    var values = [];
    for (var key in enumClass) {
        var val = parseInt(key);
        if (!isNaN(val)) values.push(val);
    }
    return values;
}

// sample usage:
var nn = getEnumNames(wijmo.DataType); // returns [ 'Object', 'String', 'Number', 'Boolean', 'Array' ]
var vv = getEnumValues(wijmo.DataType); // returns [ 0, 1, 2, 3, 4 ]
```

Globalizing Wijmo Applications

By default, Wijmo formats and parses data using the American English culture. The decimal symbol is a period, the thousand separator is a comma, and the days of the week are "Sunday" through "Saturday".

If your application targets other cultures, include references to the appropriate Wijmo culture files in your HTML pages. Wijmo includes over 40 culture files (see complete list below), and we have tools for generating new ones. If you want to target a culture that is not currently supported, contact us and we will create the file you need.

For example, to localize an application for the German culture, add this line to the head section of the page:

```
<!-- set German culture -->
<script src="http://cdn.wijmo.com/5.latest/controls/cultures/wijmo.culture.de.min.js">
</script>
```

This fiddle demonstrates the result. Notice the formatting of grid values and calendar dates.

Example: Globalization

Show me (http://jsfiddle.net/Wijmo5/wK97R)

You can edit the values in the grid to see that globalization also works for parsing data. Wijmo's Globalize class allows you to format and parse values in your applications outside of Wijmo controls as well.

Included Cultures

By default, Wijmo uses the American English culture, but for your convenience, we have included several other cultures. They are located on the CDN, but you can also find the files in your installation folder in `\Dist\controls\cultures`. The following cultures are currently included.

- **ar-AE** Arabic (United Arab Emirates)
- **eu** Basque
- **bg** Bulgarian
- **ca** Catalan
- **zh** Chinese
- **zh-HK** Chinese (Traditional, Hong Kong SAR)
- **zh-TW** Chinese (Traditional, Taiwan)
- **hr** Croatian
- **cs** Czech
- **da** Danish
- **nl** Dutch
- **en** English
- **en-CA** English (Canada)
- **en-GB** English (United Kingdom)
- **et** Estonian
- **fi** Finnish
- **fr** French
- **fr-CA** French (Canada)
- **gl** Galician
- **de** German
- **el** Greek
- **he** Hebrew
- **hi** Hindi
- **hu** Hungarian
- **id** Indonesian
- **it** Italian
- **ja** Japanese
- **kk** Kazakh
- **ko** Korean
- **lv** Latvian
- **lt** Lithuanian
- **no** Norwegian

- **pl** Polish
- **pt** Portuguese
- **ro** Romanian
- **ru** Russian
- **sr** Serbian
- **sk** Slovak
- **sl** Slovenian

es Spanish
es-419 Spanish (Latin America)
es-MX Spanish (Mexico)
sv Swedish
th Thai
tr Turkish
uk Ukrainian

Wijmo Glyphs

The **wijmo.css** file includes several glyphs defined as pure CSS. The glyphs are used by the Wijmo controls and extensions, and your applications may also use them.

Using CSS to define glyphs eliminates the need to deploy extra font or image files, and ensures the images are rendered using the foreground color and font size defined by the current theme.

To use Wijmo glyphs in your applications, add a span element to your markup and set its class to the glyph name. For example:

```
<span class="wj-glyph-diamond"></span>
```

You can use CSS to customize the appearance of the glyphs used in the Wijmo controls. For example, you could use the CSS below to hide or modify the appearance of the pencil glyph used by the **FlexGrid** to indicate rows in edit mode:

```
/* hide the pencil glyph in FlexGrid controls */
.wj-flexgrid .wj-glyph-pencil {
    display: none;
}
```

```
/* replace the pencil glyph in FlexGrid controls with a custom image */
.wj-flexgrid .wj-glyph-pencil {
    background-image:url('../images/my-pencil.png');
    background-repeat: round;
    border: 0;
    opacity: 1;
}
    .wj-flexgrid .wj-glyph-pencil:after {
        display: none;
    }
```

The table below shows the glyphs defined in the **wijmo.css** file:

Name	Glyph	Markup
asterisk		`<span class="wj-glyph-asterisk"></span>`
calendar		`<span class="wj-glyph-calendar"></span>`
check		`<span class="wj-glyph-check"></span>`
circle		`<span class="wj-glyph-circle"></span>`
clock		`<span class="wj-glyph-clock"></span>`
diamond		`<span class="wj-glyph-diamond"></span>`
down		`<span class="wj-glyph-down"></span>`
down-left		`<span class="wj-glyph-down-left"></span>`
down-right		`<span class="wj-glyph-down-right"></span>`
file		`<span class="wj-glyph-file"></span>`
filter		`<span class="wj-glyph-filter"></span>`
left		`<span class="wj-glyph-left"></span>`

Name	Glyph	Markup
minus		`<span class="wj-glyph-minus"></span>`
pencil		`<span class="wj-glyph-pencil"></span>`
plus		`<span class="wj-glyph-plus"></span>`
right		`<span class="wj-glyph-right"></span>`
square		`<span class="wj-glyph-square"></span>`
step-backward		`<span class="wj-glyph-step-backward"></span>`
step-forward		`<span class="wj-glyph-step-forward"></span>`
up		`<span class="wj-glyph-up"></span>`
up-left		`<span class="wj-glyph-up-left"></span>`
up-right		`<span class="wj-glyph-up-right"></span>`

JavaScript IntelliSense

IntelliSense is one of the best features in Visual Studio and VSCode. It saves typing effort, reduces errors, and helps you learn or remember object models as you work.

IntelliSense works great with TypeScript, but it is somewhat limited with JavaScript. You do get basic support for the language features, but external libraries are not supported unless you specify the custom definition files.

Starting with our 2016/V2 release, we are providing IntelliSense definition files for the entire Wijmo library. Simply install these files and you get full IntellliSense support when writing JavaScript code with Wijmo.

Visual Studio Installation

To install the Wijmo IntelliSense definition file in Visual Studio, you will need the "wijmo.intellisense.js" file. You can find this file in the Wijmo distribution under the "IntelliSense" folder:

```
C1Wijmo-Enterprise_5.xxxxx.xxx.zip
  Dist
    IntelliSense
      readme.txt
wijmo.intellisense.js
    Samples
      VSTemplates
```

Perform the following steps to install the Wijmo IntelliSense definition file in Visual Studio:

1. Copy the "□wijmo.intellisense.js" file to your computer
2. Open the Visual Studio settings pane at

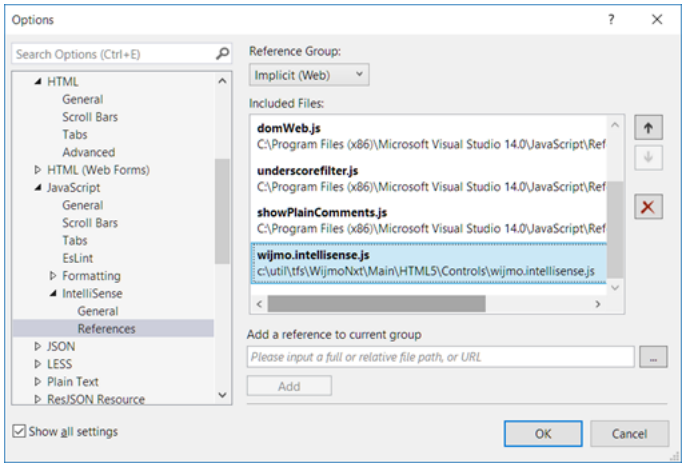
   ```
   Tools|Options|Text Editor|JavaScript|IntelliSense|References
   ```

3. In the Reference Group ComboBox, select

   ```
   Implicit (Web)
   ```

4. Add a reference to the "wijmo.intellisense.js" file
5. Click OK to apply the change

The image below shows the dialog:



VSCode Installation

To install the Wijmo IntelliSense definitions in VS Code, you will need the TypeScript definition files ("*.d.ts"). You can find these files in the Wijmo distribution under the "Controls" folder:

```
C1Wijmo-Enterprise_5.xxxxx.xxx.zip
  Dist
    controls
wijmo.d.ts
wijmo.input.d.ts
wijmo.grid.d.ts
    …
  IntelliSense
        Samples
        VSTemplates
```

Copy the "*.d.ts" files to a folder under your VSCode project root. If the project has a "jsconfig.json" configuration file, make sure the "*.d.ts" files are included. For example:

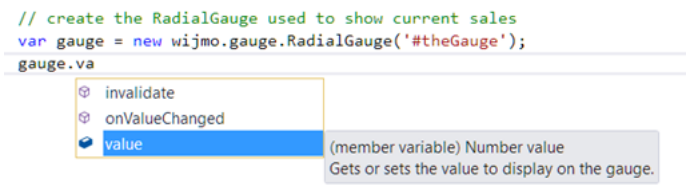
```
"files": [
  "typings/wijmo.d.ts",
  "typings/wijmo.input.d.ts",
  "typings/wijmo.grid.d.ts",
    …
]
```

Alternatively, you can specify the "*.d.ts" files by adding reference comments at the top of your "js" files. For example:

```
/// <reference path="typings/wijmo.d.ts" />
```

Using Wijmo IntelliSense

Once the Wijmo IntelliSense definition file has been installed, you will be able to search and explore the object model of any Wijmo class as you write JavaScript code. The image below shows an example:


// create the RadialGauge used to show current sales
var gauge = new wijmo.gauge.RadialGauge('#theGauge');
gauge.va
invalidate
onValueChanged
value
(member variable) Number value
Gets or sets the value to display on the gauge.

AngularJS Directives

We at ComponentOne are big fans of **AngularJS**, Google's framework for JavaScript applications. AngularJS provides templating, data-binding, MVVM, web components, and more.

One of the main advantages of AngularJS is that it supports the MVVM pattern, where the application logic is contained in Models (aka controllers, implemented in JavaScript) and the appearance is contained in Views (HTML).

To achieve this, AngularJS supports **directives**, which are custom HTML elements and attributes. AngularJS ships with a number of built-in directives, and you can easily implement your own much as you can create your own controls in WinForms or XAML applications.

Wijmo ships with AngularJS directives for all its controls. The directives are defined in the **wijmo.angular.js** file, and allow you to write code such as:

```
<div ng-app="app" ng-controller="appCtrl">

<p>This is a <b>FlexGrid</b> control:</p>
<wj-flex-grid items-source="data">
    <wj-flex-grid-column header="Country" binding="country"></wj-flex-grid-column>
    <wj-flex-grid-column header="Sales" binding="sales"></wj-flex-grid-column>
    <wj-flex-grid-column header="Expenses" binding="expenses"></wj-flex-grid-column>
    <wj-flex-grid-column header="Downloads" binding="downloads"></wj-flex-grid-column>
</wj-flex-grid>
</div>
```

Using Wijmo in AngularJS Applications

If you want to use Wijmo Angular directives in your project, add references to AngularJS, Wijmo, and then to Wijmo's AngularJS directives, as described in the Referencing Wijmo in Your Applications topic.

Once you have the references in place, tell AngularJS that your app depends on the "wj" module using code like this:

```
var app = angular.module('app', ['wj']);
```

With the app defined, you can go on to add a controller to provide data and logic as you would with any AngularJS application.

All Wijmo directives start with the "wj" prefix, followed by the control's class name using dashes instead of camel-case. The directives have attributes that match the control's properties, following the same convention.

Some of the directives support nested sub-directives. For example, the **wj-flex-grid** directive may contain one or more **wj-flex-grid-column** directives, and the **wj-flex-chart** directive may contain one or more **wj-flex-chart-series** directives. This results in a rich and expressive markup syntax that is very similar to XAML. This flexibility is essential in order to achieve the true benefits of MVVM. The appearance and layout of the controls are defined by the view; the controller only provides the data.

The fiddle below shows how this works:

Example: Angular Directives

Show me (http://jsfiddle.net/Wijmo5/QNb9X)

For more examples that use Wijmo with AngularJS, see our Demos.

Angular 2 Components

- Supports Angular version **2.2.1** or higher, including **5.*** versions.

Note This description pertains to new *external* Wijmo core library modules introduced after build 211. This is a recommended way of Wijmo interop for Angular 2 usage. A description based on *global* Wijmo core library modules can be found here.

Wijmo *components* for Angular 2 allow you to use Wijmo *controls* in Angular 2 templates markup. In terms of the TypeScript class inheritance feature, Wijmo Angular 2 *components* "extend" the *control* classes they represent. This means that when you acquire a reference to a Wijmo component, the referenced instance is a Wijmo *control* with all its properties, events and methods, and an Angular 2 *component* at the same time. A Wijmo component extends a control class and adds the necessary functionality that allows the control to be used in the Angular 2 template markup, with the fully functional one-way and two-way property bindings and event bindings. This integration is smooth, as all the players, Wijmo controls, Wijmo Angular 2 components and Angular 2 itself are written in the same TypeScript language.

Wijmo Angular 2 components are shipped as a set of modules, one module per core library module, with the "angular2" word in their names. For example, "wijmo.angular2.grid.js" module represents components for controls from the core "wijmo.grid.js" module. Module files are located in subfolders of the **NpmImages** folder of Wijmo download zip. Each subfolder provides modules in a certain module format, like CommonJS or AMD, and is effectively an npm image that can be installed into your application using "npm install <path to subfolder>" command. Refer to the accompanying readme.txt files in these folders for more details.

All Wijmo modules should be imported using their *ambient* names, which are module names prefixed with "wijmo/", and without ".js" extension. For example, this import statement imports the content of the "wijmo.angular2.grid.js" module:

```
import * as wjGrid from 'wijmo/wijmo.angular2.grid';
```

Adding Wijmo to your Angular 2 Application

Wijmo is not represented in npm registry. Instead, we ship npm images of Wijmo external modules in the **NpmImages** folder of Wijmo download zip, where the library can be installed from, using the conventional npm installation means ("npm install <path_to_folder>" command or a record in the "dependencies" option of the application's package.json file).

The NpmImages folder contains subfolders like **wijmo-commonjs-min**, **wijmo-amd-min** and **wijmo-system-min**, which are standalone npm images representing Wijmo modules in different module formats (CommonJS, AMD and System respectively).

A choice of a module format to use depends on the module loader and/or bundler tools that you use in your application. CommonJS format will work in most cases.

So, you can install Wijmo into your application using "npm install <path_to_folder>" command in NodeJS command prompt, like this:

```
npm install ../wijmo_download/NpmImages/wijmo-commonjs-min
```

This command will add the folder content to the node_modules/wijmo folder of your application.

Alternatively, you can add the following record to the package.json file of your application:

```
"dependencies": {
    "wijmo": "../wijmo_download/NpmImages/wijmo-commonjs-min",
    … other libraries
}
```

After that, each time you execute "npm install" command in your application root folder, Wijmo modules will be installed under the "node_modules" folder along with another libraries enumerated in package.json.

Importing Wijmo components

With this setup, you may import Wijmo Angular 2 modules and use the components and directives they contain. For example, this code adds a WjFlexGrid component to MyCmp component's template, with the **flex** property containing a reference to the added grid:

```
import { Component, NgModule, ViewChild } from '@angular/core';
import { WjGridModule, WjFlexGrid } from 'wijmo/wijmo.angular2.grid';

@Component({
    template: '<wj-flex-grid #flex [itemsSource]="data"></wj-flex-grid>',
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
    @ViewChild('flex') flex: WjFlexGrid;

}

@NgModule({
    imports: [WjGridModule, BrowserModule],
    declarations: [MyCmp]
})
export class MyModule {
}
```

Every Wijmo for Angular 2 JavaScript module contains an Angular 2 NgModule that exports all the components in the module. To use any of these components in your NgModule components' templates, you just need to add a reference to Wijmo NgModule to the **imports** metadata property of your NgModule decorator.

A name of NgModule is constructed from its JavaScript module name using the following schema:
Wj<JS module name without *wijmo.angular2* prefix>**Module**

For example, **WjInputModule** NgModule for **wijmo.angular2.input** JavaScript module, or **WjGridFilterModule** NgModule for **wijmo.angular2.grid.filter** JavaScript module.

Creating Wijmo controls in code

Wijmo *components* for Angular 2 are intended for a usage in templates markup. If you want to create a Wijmo control in code, you should use a Wijmo *control* from a core module for this purpose, instead of a component. A core module has the same name as a corresponding Angular 2 interop module, but without the "angular2" word in the name. For example, this code creates a FlexGrid control in code:

```
import { FlexGrid } from 'wijmo/wijmo.grid';
let flex = new FlexGrid('#host_element');
```

Note that we import FlexGrid control instead of WjFlexGrid component, and import it from the 'wijmo/wijmo.grid' module instead of 'wijmo/wijmo.angular2.grid'.

Adapting to different loader/bundler tools

Let's consider specifics of adapting Wijmo to most popular module loaders and bundlers.

WebPack

The only additional step required here is to include Wijmo css (the Dist/styles/wijmo.min.css file from Wijmo download zip) in a bundle. You should copy this file somewhere under the application's root and include it using standard WebPack means that you use for your own application's css:

```
import 'style!css!../styles/vendor/wijmo.min.css';
```

The style-loader and css-loader should be added to the "devDependencies" option of your application's package.json file:

```
"devDependencies": {
    "css-loader": "^0.23.1",
    "style-loader": "^0.13.1",
    ... another libraries
}
```

SystemJS Loader

You have to map Wijmo ambient module names to Wijmo .js files in the node_modules folder, by adding the following config options that you pass to the System.config method call:

```
map: {
    'wijmo': 'node_modules/wijmo'
},
packages: {
    'wijmo': {
        defaultExtension: 'js'
    }
}
```

Angular CLI

The only additional step required here is to include Wijmo css (the Dist/styles/wijmo.min.css file from Wijmo download zip) in a bundle. You should copy this file somewhere under the application's root and include it using standard Angular CLI means that you use for your own application's css, namely by adding a reference to this file to the application's angular-cli.json file:

```
"apps": [
    {
        "styles": ["../styles/vendor/wijmo.min.css"],
    }
],
... other options
```

Angular 2 Markup Syntax

Refer to the Angular 2 Markup Syntax topic for the description.

What Is LESS?

LESS is a dynamic style sheet language that extends CSS through the addition of variables, mixins, operations and nested rules.

- **Variables:** Reusable common values, such as color and style information.
- **Mixins:** Properties from existing styles that you use inside new styles.
- **Operations:** Mathematical creation and manipulation of CSS properties.
- **Nested rules:** CSS selectors placed inside of other CSS selectors.

LESS is also referred to as a CSS preprocessor because you can write style sheets in the extended LESS language first, and then compile them into plain CSS. LESS is open-source and you can run it on the client-side or server-side, or compile it and output it as plain CSS.

For downloads, detailed documentation, and resources, see the LESS web site: **http://lesscss.org**.

Using LESS

You can use LESS from the command line, download it as a JS file for in-browser use, or use a third party application.

Command line

You can find specific instructions and a list of command-line prompts on the LESS web site under Command Line Usage.

In-browser

1. Download a copy of less.js and save it.
2. Create a style sheet and save it using the .less file extension instead of the usual .css extension.
3. Add the following code within the <head> element of your HTML page:

```
<link rel="stylesheet/less" href="styles.less">
<script src="less.js"></script>
```

Note: In the link element, you must replace the typical rel="stylesheet" with rel="stylesheet**/less**" in order for LESS to compile in the browser. LESS style sheets must come before the LESS script.

When the page loads, **less.js** processes and compiles your LESS code live in the browser. While this is a convenient way to start developing with LESS, it is not recommended in production environments where performance is important.

Third party applications

It is advisable to compile LESS into plain CSS for production environments. If you want something more than the command line prompts offer, a wide array of third party tools is available. The LESS web site has a list of compilers and editors for various platforms and IDEs. See Online LESS Compilers.

Customizing Variables

We built Wijmo themes with LESS and make the source files available along with their compiled CSS counterparts. You can update existing themes and create new ones using the provided files.

The Wijmo themes have this file naming structure: wijmo.theme.*ThemeName*.css. To update a theme using LESS, find the corresponding wijmo.theme.*ThemeName*.less file, modify it, and re-compile it using one of the methods listed above.

Every theme is built upon a base set of colors and style options. We declare variables for these common values that we reuse throughout the theme. Note that LESS variables start with the @ symbol.

```
@background: #f3f3f3;
@header: #54443b;
@primary: #2780ec;
@text: #26211f;
@button: #5f534c;
@tool-tip: #e5d9cf;
@grid-cell-border: true;
@grid-right-side-col: none;
@border-radius: 4px;
@background-grad: false;
@button-grad: false;
@header-grad: false;
```

In addition to base variables, we use import statements to include additional LESS files to use as mixins. These contain color functions, style mixins, and mixin guards.

```
@import "mixins/color-functions";
@import "mixins/guards";
@import "mixins/styles";
@import "mixins/chart";
```

The color functions generate a wider pallet of additional colors based on the initial colors selected and the mixin guards are designed to ensure contrast. The mixin guards evaluate colors based on their lightness and write CSS accordingly: If an area's background color is greater than 50% in lightness, the text in that area renders in a darker color, and vice versa.

Creating New Themes

You can create new themes by duplicating an existing theme and then modifying the base set of colors and style options. Here is a key to the color and style variables.

@background - Background color for the control.
@header - Background color for control headers (e.g. FlexGrid, FlexChart).
@primary - Primary accent color used throughout the theme.
@text - Default text color.
@button - Background color for buttons.
@tool-tip - Background color for tooltips.
@border-radius - Global border radius setting for all controls.
@grid-cell-border - Toggle for FlexGrid cell borders. Set to 'false' to turn off all grid borders.
@grid-left-side-col - Background color for the FlexGrid's row header cells (same as @header by default).
@background-grad - Toggle for control background color gradient. Set to 'true' to add a gradient.
@button-grad - Toggle for button background color gradient. Set to 'true' to add a gradient.
@header-grad - Toggle for header background color gradient. Set to 'true' to add a gradient.

Deploying New Themes

Once you create a new theme and output it as plain CSS, place it in the styles directory and include it in your project. Now you can link to it from any HTML page that contains Wijmo controls.

```
<link href="styles/NewTheme.css" rel="stylesheet">
```

wijmo Module

File
`wijmo.js`
Module
wijmo

Contains utilities used by all controls and modules, as well as the **Control** and **Event** classes.

Classes

- Binding
- CancelEventArgs
- Clipboard
- Color
- Control
- DateTime
- Event
- EventArgs
- Globalize
- Point
- PrintDocument
- PropertyChangedEventArgs
- Rect
- RequestErrorEventArgs
- Size
- Tooltip
- TooltipEventArgs

Interfaces

- IEventHandler
- IQueryInterface

Enums

- Aggregate
- DataType
- Key

Properties

- culture

Methods

- addClass
- animate
- asArray
- asBoolean
- asCollectionView
- asDate
- asEnum
- asFunction
- asInt
- asNumber
- assert
- asString
- asType
- changeType
- clamp
- closest
- closestClass
- contains
- copy
- createElement
- enable
- escapeHtml
- format
- getActiveElement
- getAggregate
- getElement
- getElementRect
- getType
- getUniqueId
- getVersion
- hasClass
- hasItems
- hidePopup
- httpRequest
- isArray
- isBoolean
- isDate
- isFunction
- isInt
- isNullOrWhiteSpace
- isNumber
- isObject
- isPrimitive
- isString
- isUndefined
- mouseToPage
- moveFocus
- removeClass

- setAttribute
- setCss
- setSelectionRange
- setText
- showPopup
- toFixed
- toggleClass
- toHeaderCase
- tryCast

Properties

- **culture**

Gets or sets an object that contains all localizable strings in the Wijmo library.

The culture selector is a two-letter string that represents an **`ISO 639 culture (http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes)`**.

Type
any

Methods

- **addClass**

```
addClass(e: Element, className: string): void
```

Adds a class to an element.

Parameters

- **e**: **Element**
  Element that will have the class added.
- **className**: **string**
  Class to add to the element.

Returns
void

animate

```
animate(apply: Function, duration?: Control, step?: Control): any
```

Calls a function on a timer with a parameter varying between zero and one.

Use this function to create animations by modifying document properties or styles on a timer.

For example, the code below changes the opacity of an element from zero to one in one second:

```
var element = document.getElementById('someElement');
animate(function(pct) {
  element.style.opacity = pct;
}, 1000);
```

The function returns an interval ID that you can use to stop the animation. This is typically done when you are starting a new animation and wish to suspend other on-going animations on the same element. For example, the code below keeps track of the interval ID and clears if before starting a new animation:

```
var element = document.getElementById('someElement');
if (this._animInterval) {
  clearInterval(this._animInterval);
}
var self = this;
self._animInterval = animate(function(pct) {
  element.style.opacity = pct;
  if (pct == 1) {
    self._animInterval = null;
  }
}, 1000);
```

Parameters

- **apply**: **Function**
  Callback function that modifies the document. The function takes a single parameter that represents a percentage.
- **duration**: **Control** OPTIONAL
  The duration of the animation, in milliseconds.
- **step**: **Control** OPTIONAL
  The interval between animation frames, in milliseconds.

Returns
any

asArray

```
asArray(value: any, nullOK?: boolean): any[]
```

Asserts that a value is an array.

Parameters

- **value**: **any**
  Value supposed to be an array.

- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
any[]

asBoolean

```
asBoolean(value: boolean, nullOK?: boolean): boolean
```

Asserts that a value is a Boolean.

Parameters

- **value**: **boolean**
  Value supposed to be Boolean.

- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
boolean

asCollectionView

```
asCollectionView(value: any, nullOK?: boolean): ICollectionView
```

Asserts that a value is an **`ICollectionView`** or an Array.

Parameters

- **value**: **any**
  Array or **`ICollectionView`**.

- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
`ICollectionView`

asDate

```
asDate(value: Date, nullOK?: boolean): Date
```

Asserts that a value is a Date.

Parameters

- **value**: **Date**
  Value supposed to be a Date.

- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
Date

asEnum

```
asEnum(value: number, enumType: any, nullOK?: boolean): number
```

Asserts that a value is a valid setting for an enumeration.

Parameters

- **value**: **number**
  Value supposed to be a member of the enumeration.
- **enumType**: **any**
  Enumeration to test for.
- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
number

asFunction

```
asFunction(value: any, nullOK?: boolean): Function
```

Asserts that a value is a function.

Parameters

- **value**: **any**
  Value supposed to be a function.
- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
Function

asInt

```
asInt(value: number, nullOK?: boolean, positive?: boolean): number
```

Asserts that a value is an integer.

Parameters

- **value**: **number**
  Value supposed to be an integer.
- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.
- **positive**: **boolean** OPTIONAL
  Whether to accept only positive integers.

Returns
number

asNumber

```
asNumber(value: number, nullOK?: boolean, positive?: boolean): number
```

Asserts that a value is a number.

Parameters

- **value**: **number**
  Value supposed to be numeric.
- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.
- **positive**: **boolean** OPTIONAL
  Whether to accept only positive numeric values.

Returns
number

assert

```
assert(condition: boolean, msg: string): void
```

Throws an exception if a condition is false.

Parameters

- **condition**: **boolean**
  Condition expected to be true.

- **msg**: **string**
  Message of the exception if the condition is not true.

Returns
void

asString

```
asString(value: string, nullOK?: boolean): string
```

Asserts that a value is a string.

Parameters

- **value**: **string**
  Value supposed to be a string.

- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
string

asType

```
asType(value: any, type: any, nullOK?: boolean): any
```

Asserts that a value is an instance of a given type.

Parameters

- **value**: **any**
  Value to be checked.
- **type**: **any**
  Type of value expected.
- **nullOK**: **boolean** OPTIONAL
  Whether null values are acceptable.

Returns
any

changeType

```
changeType(value: any, type: DataType, format: string): any
```

Changes the type of a value.

If the conversion fails, the original value is returned. To check if a conversion succeeded, you should check the type of the returned value.

Parameters

- **value**: **any**
  Value to convert.
- **type**: **DataType**
  DataType to convert the value to.
- **format**: **string**
  Format to use when converting to or from strings.

Returns
any

clamp

```
clamp(value: number, min: number, max: number): number
```

Clamps a value between a minimum and a maximum.

Parameters

- **value**: **number**
  Original value.
- **min**: **number**
  Minimum allowed value.
- **max**: **number**
  Maximum allowed value.

Returns
number

closest

```
closest(e: any, selector: string): Node
```

Finds the closest ancestor (including the original element) that satisfies a selector.

Parameters

- **e**: **any**
  Element where the search should start.
- **selector**: **string**
  A string containing a selector expression to match elements against.

Returns
Node

closestClass

```
closestClass(e: any, className: string): Node
```

Finds the closest ancestor (including the original element) that satisfies a class selector.

Parameters

- **e**: **any**
  Element where the search should start.

- **className**: **string**
  A string containing the class name to match elements against.

Returns
Node

contains

```
contains(parent: any, child: any): boolean
```

Checks whether an HTML element contains another.

Parameters

- **parent**: **any**
  Parent element.

- **child**: **any**
  Child element.

Returns
boolean

copy

```
copy(dst: any, src: any): void
```

Copies properties from an object to another.

This method is typically used to initialize controls and other Wijmo objects by setting their properties and assigning event handlers.

The destination object must define all the properties defined in the source, or an error will be thrown.

Parameters

- **dst**: **any**
  The destination object.

- **src**: **any**
  The source object.

Returns
void

createElement

```
createElement(html: string, appendTo?: HTMLElement): HTMLElement
```

Creates an element from an HTML string.

Parameters

- **html**: **string**
  HTML fragment to convert into an HTMLElement.

- **appendTo**: **HTMLElement** OPTIONAL
  Optional HTMLElement to append the new element to.

Returns
HTMLElement

enable

```
enable(e: HTMLElement, value: boolean): void
```

Enables or disables an element.

Parameters

- **e**: **HTMLElement**
  Element to enable or disable.

- **value**: **boolean**
  Whether to enable or disable the element.

Returns
void

escapeHtml

```
escapeHtml(text: string): void
```

Escapes a string by replacing HTML characters as text entities.

Strings entered by uses should always be escaped before they are displayed in HTML pages. This ensures page integrity and prevents HTML/javascript injection attacks.

Parameters

- **text**: **string**
  Text to escape.

Returns
void

format

```
format(format: string, data: any, formatFunction?: Function): string
```

Replaces each format item in a specified string with the text equivalent of an object's value.

The function works by replacing parts of the **formatString** with the pattern '{name:format}' with properties of the **data** parameter. For example:

```
var data = { name: 'Joe', amount: 123456 };
var msg = wijmo.format('Hello {name}, you won {amount:n2}!', data);
```

The **format** function supports pluralization. If the format string is a JSON-encoded object with 'count' and 'when' properties, the method uses the 'count' parameter of the data object to select the appropriate format from the 'when' property. For example:

```
var fmt = {
    count: 'count',
    when: {
        0: 'No items selected.',
        1: 'One item is selected.',
        2: 'A pair is selected.',
        'other': '{count:n0} items are selected.'
    }
}
fmt = JSON.stringify(fmt);
console.log(wijmo.format(fmt, { count: 0 })); // No items selected.
console.log(wijmo.format(fmt, { count: 1 })); // One item is selected.
console.log(wijmo.format(fmt, { count: 2 })); // A pair is selected.
console.log(wijmo.format(fmt, { count: 12 })); 12 items are selected.
```

The optional **formatFunction** allows you to customize the content by providing context-sensitive formatting. If provided, the format function gets called for each format element and gets passed the data object, the parameter name, the format, and the value; it should return an output string. For example:

```
var data = { name: 'Joe', amount: 123456 };
var msg = wijmo.format('Hello {name}, you won {amount:n2}!', data,
    function (data, name, fmt, val) {
        if (wijmo.isString(data[name])) {
            val = wijmo.escapeHtml(data[name]);
        }
        return val;
    }
);
```

Parameters

- **format**: **string**
  A composite format string.
- **data**: **any**

The data object used to build the string.

- **formatFunction**: **Function** OPTIONAL
  An optional function used to format items in context.

Returns
string

getActiveElement

```
getActiveElement(): void
```

Gets a reference to the element that contains the focus, accounting for shadow document fragments.

Returns
void

getAggregate

```
getAggregate(aggType: Aggregate, items: any[], binding?: string): void
```

Calculates an aggregate value from the values in an array.

Parameters

- **aggType**: **Aggregate**
  Type of aggregate to calculate.
- **items**: **any**[]
  Array with the items to aggregate.
- **binding**: **string** OPTIONAL
  Name of the property to aggregate on (in case the items are not simple values).

Returns
void

getElement

```
getElement(selector: any): HTMLElement
```

Gets an element from a jQuery-style selector.

Parameters

- **selector**: **any**
  An element, a query selector string, or a jQuery object.

Returns
HTMLElement

getElementRect

```
getElementRect(e: Element): Rect
```

Gets the bounding rectangle of an element in page coordinates.

This is similar to the **getBoundingClientRect** function, except that uses viewport coordinates, which change when the document scrolls.

Parameters

- **e**: **Element**

Returns
Rect

getType

```
getType(value: any): DataType
```

Gets the type of a value.

Parameters

- **value**: **any**
  Value to test.

Returns
DataType

getUniqueId

```
getUniqueId(baseId: string): string
```

Creates a new unique id for an element by adding sequential numbers to a given base id.

Parameters

- **baseId**: **string**
  String to use as a basis for generating the unique id.

Returns
string

getVersion

```
getVersion(): string
```

Gets the version of the Wijmo library that is currently loaded.

Returns
string

hasClass

```
hasClass(e: Element, className: string): boolean
```

Checks whether an element has a class.

Parameters

- **e**: **Element**
  Element to check.

- **className**: **string**
  Class to check for.

Returns
boolean

hasItems

```
hasItems(value: ICollectionView): boolean
```

Checks whether an **ICollectionView** is defined and not empty.

Parameters

- **value**: **ICollectionView**
  ICollectionView to check.

Returns
boolean

hidePopup

```
hidePopup(popup: HTMLElement, remove?: boolean, fadeOut?: boolean): any
```

Hides a popup element previously displayed with the **showPopup** method.

Parameters

- **popup**: **HTMLElement**
  Popup element to hide.
- **remove**: **boolean** OPTIONAL
  Whether to remove the popup from the DOM or just to hide it.
- **fadeOut**: **boolean** OPTIONAL
  Whether to use a fade-out animation to make the popup disappear gradually.

Returns
any

httpRequest

```
httpRequest(url: string, settings?: any): XMLHttpRequest
```

Performs HTTP requests.

Parameters

- **url**: **string**
  String containing the URL to which the request is sent.

- **settings**: **any** OPTIONAL
  An optional object used to configure the request.

  The **settings** object may contain the following:

method	The HTTP method to use for the request (e.g. "POST", "GET", "PUT"). The default is "GET".
data	Data to be sent to the server. It is appended to the url for GET requests, and converted to a string for other requests.
async	By default, all requests are sent asynchronously (i.e. this is set to true by default). If you need synchronous requests, set this option to false.
success	A function to be called if the request succeeds. The function gets passed a single parameter of type **XMLHttpRequest**.
error	A function to be called if the request fails. The function gets passed a single parameter of type **XMLHttpRequest**.
complete	A function to be called when the request finishes (after success and error callbacks are executed). The function gets passed a single parameter of type **XMLHttpRequest**.
beforeSend	A function to be called immediately before the request us sent. The function gets passed a single parameter of type **XMLHttpRequest**.
requestHeaders	A JavaScript object containing key/value pairs to be added to the request headers.
user	A username to be used with **XMLHttpRequest** in response to an HTTP access authentication request.
password	A password to be used with **XMLHttpRequest** in response to an HTTP access authentication request.

  Use the **success** to obtain the result of the request which is provided in the callback's **XMLHttpRequest** parameter. For example, the code below uses the **`httpRequest`** method to retrieve a list of customers from an OData service:

```
wijmo.httpRequest('http://services.odata.org/Northwind/Northwind.svc/Customers?$format=json', {
  success: function (xhr) {
    var response = JSON.parse(xhr.response),
        customers = response.value;

    // do something with the customers...
  }
});
```

Returns
XMLHttpRequest

isArray

```
isArray(value: any): boolean
```

Determines whether an object is an Array.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isBoolean

```
isBoolean(value: any): boolean
```

Determines whether an object is a Boolean.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isDate

```
isDate(value: any): boolean
```

Determines whether an object is a Date.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isFunction

```
isFunction(value: any): boolean
```

Determines whether an object is a function.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isInt

```
isInt(value: any): boolean
```

Determines whether an object is an integer.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isNullOrWhiteSpace

```
isNullOrWhiteSpace(value: string): boolean
```

Determines whether a string is null, empty, or whitespace only.

Parameters

- **value**: **string**
  Value to test.

Returns
boolean

isNumber

```
isNumber(value: any): boolean
```

Determines whether an object is a number.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isObject

```
isObject(value: any): boolean
```

Determines whether a value is an object (as opposed to a value type, an array, or a Date).

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isPrimitive

```
isPrimitive(value: any): boolean
```

Determines whether an object is a primitive type (string, number, Boolean, or Date).

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isString

```
isString(value: any): boolean
```

Determines whether an object is a string.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

isUndefined

```
isUndefined(value: any): boolean
```

Determines whether an object is undefined.

Parameters

- **value**: **any**
  Value to test.

Returns
boolean

mouseToPage

```
mouseToPage(e: any): Point
```

Converts mouse or touch event arguments into a **`Point`** in page coordinates.

Parameters

- **e**: **any**

Returns
`Point`

moveFocus

```
moveFocus(parent: HTMLElement, offset: number): void
```

Moves the focus to the next/previous/first focusable child within a given parent element.

Parameters

- **parent**: **HTMLElement**
  Parent element.

- **offset**: **number**
  Offset to use when moving the focus (use zero to focus on the first focusable child).

Returns
void

removeClass

```
removeClass(e: Element, className: string): void
```

Removes a class from an element.

Parameters

- **e**: **Element**
  Element that will have the class removed.

- **className**: **string**
  Class to remove from the element.

Returns
void

setAttribute

```
setAttribute(e: Element, name: string, value?: any, keep?: boolean): void
```

Sets or clears an attribute on an element.

Parameters

- **e**: **Element**
  Element that will be updated.
- **name**: **string**
  Name of the attribute to add or remove.
- **value**: **any** OPTIONAL
  Value of the attribute, or null to remove the attribute from the element.
- **keep**: **boolean** OPTIONAL
  Whether to keep original attribute if present.

Returns
void

setCss

```
setCss(e: any, css: any): void
```

Modifies the style of an element by applying the properties specified in an object.

Parameters

- **e**: **any**
  Element or array of elements whose style will be modified.
- **css**: **any**
  Object containing the style properties to apply to the element.

Returns
void

setSelectionRange

```
setSelectionRange(e: HTMLInputElement, start: number, end?: number): void
```

Sets the start and end positions of a selection in a text field.

This method is similar to the native **`setSelectionRange`** method in HTMLInputElement objects, except it checks for conditions that may cause exceptions (element not in the DOM, disabled, or hidden).

Parameters

- **e**: **HTMLInputElement**
- **start**: **number**
  Offset into the text field for the start of the selection.
- **end**: **number** OPTIONAL
  Offset into the text field for the end of the selection.

Returns
void

setText

```
setText(e: HTMLElement, text: string): void
```

Sets the text content of an element.

Parameters

- **e**: **HTMLElement**
  Element that will have its content updated.
- **text**: **string**
  Plain text to be assigned to the element.

Returns
void

showPopup

```
showPopup(popup: HTMLElement, ref?: any, above?: boolean, fadeIn?: boolean, copyStyles?: boolean): any
```

Shows an element as a popup.

The popup element becomes a child of the body element, and is positioned above or below a reference rectangle, depending on how much room is available.

The reference rectangle may be specified as one of the following:

HTMLElement
The bounding rectangle of the element.
MouseEvent
The bounding rectangle of the event's target element.
Rect
The given rectangle.
null
No reference rectangle; the popup is centered on the window.

Call the **`hidePopup`** method to hide the popup.

Parameters

- **popup**: **HTMLElement**
  Element to show as a popup.

- **ref**: **any** OPTIONAL
  Reference element or rectangle used to position the popup.

- **above**: **boolean** OPTIONAL
  Position popup above the reference rectangle if possible.

- **fadeIn**: **boolean** OPTIONAL
  Use a fade-in animation to make the popup appear gradually.

- **copyStyles**: **boolean** OPTIONAL
  Copy font and color styles from reference element.

Returns
any

toFixed

```
toFixed(value: number, prec: number, truncate: boolean): number
```

Rounds or truncates a number to a specified precision.

Parameters

- **value**: **number**
  Value to round or truncate.
- **prec**: **number**
  Number of decimal digits for the result.
- **truncate**: **boolean**
  Whether to truncate or round the original value.

Returns
number

toggleClass

```
toggleClass(e: Element, className: string, addOrRemove?: boolean): void
```

Adds or removes a class to or from an element.

Parameters

- **e**: **Element**
  Element that will have the class added.
- **className**: **string**
  Class to add or remove.
- **addOrRemove**: **boolean** OPTIONAL
  Whether to add or remove the class. If not provided, the class is toggled. Use true to add class to element and false to remove class from element.

Returns
void

toHeaderCase

```
toHeaderCase(text: string): string
```

Converts a camel-cased string into a header-type string by capitalizing the first letter and adding spaces before uppercase characters preceded by lower-case characters.

For example, 'somePropertyName' becomes 'Some Property Name'.

Parameters

- **text**: **string**
  String to convert to header case.

Returns
string

tryCast

```
tryCast(value: any, type: any): any
```

Casts a value to a type if possible.

Parameters

- **value**: **any**
  Value to cast.

- **type**: **any**
  Type or interface name to cast to.

Returns
any

Binding Class

File
`wijmo.js`
Module
wijmo

Provides binding to complex properties (e.g. 'customer.address.city')

Constructor

- constructor

Properties

- path

Methods

- getValue
- setValue

Constructor

constructor

```
constructor(path: string): Binding
```

Initializes a new instance of the **Binding** class.

Parameters

- **path**: **string**
  Name of the property to bind to.

Returns
Binding

Properties

path

Gets or sets the path for the binding.

In the simplest case, the path is the name of the property of the source object to use for the binding (e.g. 'street').

Sub-properties of a property can be specified by a syntax similar to that used in JavaScript (e.g. 'address.street').

Type
string

Methods

getValue

```
getValue(object: any): any
```

Gets the binding value for a given object.

If the object does not contain the property specified by the binding **`path`**, the method returns null.

Parameters

- **object**: **any**
  The object that contains the data to be retrieved.

Returns
any

setValue

```
setValue(object: any, value: any): void
```

Sets the binding value on a given object.

If the object does not contain the property specified by the binding **`path`**, the value is not set.

Parameters

- **object**: **any**
  The object that contains the data to be set.

- **value**: **any**
  Data value to set.

Returns
void

CancelEventArgs Class

File
`wijmo.js`
Module
wijmo
Base Class
EventArgs
Derived Classes
RequestErrorEventArgs, **TooltipEventArgs**, **PageChangingEventArgs**, **CellRangeEventArgs**, **RenderEventArgs**, **TreeNodeDragDropEventArgs**, **TreeNodeEventArgs**

Provides arguments for cancellable events.

Properties

- cancel
- empty

Properties

- **cancel**

Gets or sets a value that indicates whether the event should be canceled.

Type
boolean

- STATIC **empty**

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

Clipboard Class

File
wijmo.js
Module
wijmo

Static class that provides utility methods for clipboard operations.

The **Clipboard** class provides static **copy** and **paste** methods that can be used by controls to customize the clipboard content during clipboard operations.

For example, the code below shows how a control could intercept the clipboard shortcut keys and provide custom clipboard handling:

```
rootElement.addEventListener('keydown', function(e) {

  // copy: ctrl+c or ctrl+Insert
  if (e.ctrlKey && (e.keyCode == 67 || e.keyCode == 45)) {
    var text = this.getClipString();
    Clipboard.copy(text);
    return;
  }

  // paste: ctrl+v or shift+Insert
  if ((e.ctrlKey && e.keyCode == 86) || (e.shiftKey && e.keyCode == 45)) {
    Clipboard.paste(function (text) {
      this.setClipString(text);
    });
    return;
  }
});
```

Methods

- copy
- paste

Methods

STATIC copy

```
copy(text: string): void
```

Copies a string to the clipboard.

This method only works if invoked immediately after the user pressed a clipboard copy command (such as ctrl+c).

Parameters

- **text**: **string**
  Text to copy to the clipboard.

Returns
void

STATIC paste

```
paste(callback: Function): void
```

Gets a string from the clipboard.

This method only works if invoked immediately after the user pressed a clipboard paste command (such as ctrl+v).

Parameters

- **callback**: **Function**
  Function called when the clipboard content has been retrieved. The function receives the clipboard content as a parameter.

Returns
void

Color Class

File
`wijmo.js`
Module
wijmo

Represents a color.

The **Color** class parses colors specified as CSS strings and exposes their red, green, blue, and alpha channels as read-write properties.

The **Color** class also provides **fromHsb** and **fromHsl** methods for creating colors using the HSB and HSL color models instead of RGB, as well as **getHsb** and **getHsl** methods for retrieving the color components using those color models.

Finally, the **Color** class provides an **interpolate** method that creates colors by interpolating between two colors using the HSL model. This method is especially useful for creating color animations with the **animate** method.

Constructor

- constructor

Properties

- a
- b
- g
- r

Methods

- equals
- fromHsb
- fromHsl
- fromRgba
- fromString
- getHsb
- getHsl
- interpolate
- toOpaque
- toString

Constructor

constructor

```
constructor(color: string): Color
```

Initializes a new **Color** from a CSS color specification.

Parameters

- **color**: **string**
  CSS color specification.

Returns
Color

Properties

a

Gets or sets the alpha component of this **Color**, in a range from 0 to 1 (zero is transparent, one is solid).

Type
number

b

Gets or sets the blue component of this **Color**, in a range from 0 to 255.

Type
number

g

Gets or sets the green component of this **Color**, in a range from 0 to 255.

Type
number

r

Gets or sets the red component of this **Color**, in a range from 0 to 255.

Type
number

Methods

equals

```
equals(clr: Color): boolean
```

Returns true if a **`Color`** has the same value as this **`Color`**.

Parameters

- **clr**: **Color**
  `Color` to compare to this **`Color`**.

Returns
boolean

STATIC fromHsb

```
fromHsb(h: number, s: number, b: number, a?: number): Color
```

Creates a new **`Color`** using the specified HSB values.

Parameters

- **h**: **number**
  Hue value, from 0 to 1.
- **s**: **number**
  Saturation value, from 0 to 1.
- **b**: **number**
  Brightness value, from 0 to 1.
- **a**: **number** OPTIONAL
  Alpha value, from 0 to 1.

Returns
`Color`

STATIC fromHsl

```
fromHsl(h: number, s: number, l: number, a?: number): Color
```

Creates a new **`Color`** using the specified HSL values.

Parameters

- **h**: **number**
  Hue value, from 0 to 1.
- **s**: **number**
  Saturation value, from 0 to 1.
- **l**: **number**
  Lightness value, from 0 to 1.
- **a**: **number** OPTIONAL
  Alpha value, from 0 to 1.

Returns
`Color`

STATIC fromRgba

```
fromRgba(r: number, g: number, b: number, a?: number): Color
```

Creates a new **`Color`** using the specified RGBA color channel values.

Parameters

- **r**: **number**
  Value for the red channel, from 0 to 255.
- **g**: **number**
  Value for the green channel, from 0 to 255.
- **b**: **number**
  Value for the blue channel, from 0 to 255.
- **a**: **number** OPTIONAL
  Value for the alpha channel, from 0 to 1.

Returns
`Color`

STATIC fromString

```
fromString(value: string): Color
```

Creates a new **`Color`** from a CSS color string.

Parameters

- **value**: **string**
  String containing a CSS color specification.

Returns
`Color`

getHsb

```
getHsb(): number[]
```

Gets an array with this color's HSB components.

Returns
number[]

getHsl

```
getHsl(): number[]
```

Gets an array with this color's HSL components.

Returns
number[]

STATIC interpolate

```
interpolate(c1: Color, c2: Color, pct: number): Color
```

Creates a **`Color`** by interpolating between two colors.

Parameters

- **c1**: **Color**
  First color.
- **c2**: **Color**
  Second color.
- **pct**: **number**
  Value between zero and one that determines how close the interpolation should be to the second color.

Returns
`Color`

STATIC toOpaque

```
toOpaque(c: any, bkg?: any): Color
```

Gets the closest opaque color to a given color.

Parameters

- **c**: **any**
  Color to be converted to an opaque color (the color may also be specified as a string).

- **bkg**: **any** OPTIONAL
  Background color to use when removing the transparency (defaults to white).

Returns
Color

toString

```
toString(): string
```

Gets a string representation of this **Color**.

Returns
string

Control Class

File
`wijmo.js`
Module
wijmo
Derived Classes
FlexGrid, **ColumnFilterEditor**, **ConditionFilterEditor**, **ValueFilterEditor**, **GroupPanel**, **Calendar**, **ColorPicker**, **DropDown**, **InputMask**, **InputNumber**, **ListBox**, **Popup**, **FlexChartBase**, **Gauge**, **TreeView**, **DetailDialog**, **PivotChart**, **PivotFieldEditor**, **PivotFilterEditor**, **PivotPanel**, **ViewerBase**

Base class for all Wijmo controls.

The **Control** class handles the association between DOM elements and the actual control. Use the **hostElement** property to get the DOM element that is hosting a control, or the **getControl** method to get the control hosted in a given DOM element.

The **Control** class also provides a common pattern for invalidating and refreshing controls, for updating the control layout when its size changes, and for handling the HTML templates that define the control structure.

Constructor

- constructor

Properties

- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, options?, invalidateOnResize?: boolean): Control
```

Initializes a new instance of the **`Control`** class and attaches it to a DOM element.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.
- **invalidateOnResize**: **boolean** OPTIONAL
  Whether the control should be invalidated when it is resized.

Returns
Control

Properties

hostElement

Gets the DOM element that is hosting the control.

Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **InputNumber** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **`invalidateAll`** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Arguments
`EventArgs`

DateTime Class

File
`wijmo.js`
Module
wijmo

Provides date and time utilities.

Methods

- addDays
- addHours
- addMinutes
- addMonths
- addSeconds
- addYears
- clone
- equals
- fromDateTime
- fromFiscal
- newDate
- sameDate
- sameTime
- toFiscal

Methods

STATIC addDays

```
addDays(value: Date, days: number): Date
```

Gets a new Date that adds the specified number of days to a given Date.

Parameters

- **value**: **Date**
  Original date.
- **days**: **number**
  Number of days to add to the given date.

Returns
Date

STATIC addHours

```
addHours(value: Date, hours: number): Date
```

Gets a new Date that adds the specified number of hours to a given Date.

Parameters

- **value**: **Date**
  Original date.

- **hours**: **number**
  Number of hours to add to the given date.

Returns
Date

STATIC addMinutes

```
addMinutes(value: Date, minutes: number): Date
```

Gets a new Date that adds the specified number of minutes to a given Date.

Parameters

- **value**: **Date**
  Original date.

- **minutes**: **number**
  Number of minutes to add to the given date.

Returns
Date

STATIC addMonths

```
addMonths(value: Date, months: number): Date
```

Gets a new Date that adds the specified number of months to a given Date.

Parameters

- **value**: **Date**
  Original date.

- **months**: **number**
  Number of months to add to the given date.

Returns
Date

STATIC addSeconds

```
addSeconds(value: Date, seconds: number): Date
```

Gets a new Date that adds the specified number of seconds to a given Date.

Parameters

- **value**: **Date**
  Original date.

- **seconds**: **number**
  Number of seconds to add to the given date.

Returns
Date

STATIC addYears

```
addYears(value: Date, years: number): Date
```

Gets a new Date that adds the specified number of years to a given Date.

Parameters

- **value**: **Date**
  Original date.

- **years**: **number**
  Number of years to add to the given date.

Returns
Date

STATIC clone

```
clone(date: Date): Date
```

Creates a copy of a given Date object.

Parameters

- **date**: **Date**
  Date object to copy.

Returns
Date

STATIC equals

```
equals(d1: Date, d2: Date): boolean
```

Returns true if two Date objects refer to the same date and time.

Parameters

- **d1**: **Date**
  First date.

- **d2**: **Date**
  Second date.

Returns
boolean

STATIC fromDateTime

```
fromDateTime(date: Date, time: Date): Date
```

Gets a Date object with the date and time set on two Date objects.

Parameters

- **date**: **Date**
  Date object that contains the date (day/month/year).

- **time**: **Date**
  Date object that contains the time (hour:minute:second).

Returns
Date

STATIC fromFiscal

```
fromFiscal(date: Date, govt: boolean): void
```

Converts a fiscal year date to a calendar date using the current culture.

Parameters

- **date**: **Date**
  Fiscal year date.

- **govt**: **boolean**
  Whether to use the government or corporate fiscal year.

Returns
void

STATIC newDate

```
newDate(year?: number, month?: number, day?: number, hour?: number, min?: number, sec?: number, ms?: number): Date
```

Gets a new Date object instance.

Parameters

- **year**: **number** OPTIONAL
  Integer value representing the year, defaults to current year.
- **month**: **number** OPTIONAL
  Integer value representing the month (0-11), defaults to current month.
- **day**: **number** OPTIONAL
  Integer value representing the day (1-31), defaults to current day.
- **hour**: **number** OPTIONAL
  Integer value representing the hour, defaults to zero.
- **min**: **number** OPTIONAL
  Integer value representing the minute, defaults to zero.
- **sec**: **number** OPTIONAL
  Integer value representing the second, defaults to zero.
- **ms**: **number** OPTIONAL
  Integer value representing the millisecond, defaults to zero.

Returns
Date

STATIC sameDate

```
sameDate(d1: Date, d2: Date): boolean
```

Returns true if two Date objects refer to the same date (ignoring time).

Parameters

- **d1**: **Date**
  First date.

- **d2**: **Date**
  Second date.

Returns
boolean

STATIC sameTime

```
sameTime(d1: Date, d2: Date): boolean
```

Returns true if two Date objects refer to the same time (ignoring date).

Parameters

- **d1**: **Date**
  First date.

- **d2**: **Date**
  Second date.

Returns
boolean

STATIC toFiscal

```
toFiscal(date: Date, govt: boolean): void
```

Converts a calendar date to a fiscal date using the current culture.

Parameters

- **date**: **Date**
  Calendar date.

- **govt**: **boolean**
  Whether to use the government or corporate fiscal year.

Returns
void

Event Class

File
`wijmo.js`
Module
wijmo

Represents an event.

Wijmo events are similar to .NET events. Any class may define events by declaring them as fields. Any class may subscribe to events using the event's **addHandler** method and unsubscribe using the **removeHandler** method.

Wijmo event handlers take two parameters: *sender* and *args*. The first is the object that raised the event, and the second is an object that contains the event parameters.

Classes that define events follow the .NET pattern where for every event there is an *on[EVENTNAME]* method that raises the event. This pattern allows derived classes to override the *on[EVENTNAME]* method and handle the event before and/or after the base class raises the event. Derived classes may even suppress the event by not calling the base class implementation.

For example, the TypeScript code below overrides the **onValueChanged** event for a control to perform some processing before and after the **valueChanged** event fires:

```
// override base class
onValueChanged(e: EventArgs) {

  // execute some code before the event fires
  console.log('about to fire valueChanged');

  // optionally, call base class to fire the event
  super.onValueChanged(e);

  // execute some code after the event fired
  console.log('valueChanged event just fired');
}
```

Properties

- hasHandlers

Methods

- addHandler
- raise
- removeAllHandlers
- removeHandler

Properties

hasHandlers

Gets a value that indicates whether this event has any handlers.

Type
boolean

Methods

addHandler

```
addHandler(handler: IEventHandler, self?: any): void
```

Adds a handler to this event.

Parameters

- **handler**: **IEventHandler**
  Function invoked when the event is raised.
- **self**: **any** OPTIONAL
  Object that defines the event handler (accessible as 'this' from the handler code).

Returns
void

raise

```
raise(sender: any, args?: EventArgs): void
```

Raises this event, causing all associated handlers to be invoked.

Parameters

- **sender**: **any**
  Source object.
- **args**: **EventArgs** OPTIONAL
  Event parameters.

Returns
void

removeAllHandlers

```
removeAllHandlers(): void
```

Removes all handlers associated with this event.

Returns
void

removeHandler

```
removeHandler(handler: IEventHandler, self?: any): void
```

Removes a handler from this event.

Parameters

- **handler**: **IEventHandler**
  Function invoked when the event is raised.

- **self**: **any** OPTIONAL
  Object that defines the event handler (accessible as 'this' from the handler code).

Returns
void

EventArgs Class

File
`wijmo.js`
Module
wijmo
Derived Classes
CancelEventArgs, **PropertyChangedEventArgs**, **NotifyCollectionChangedEventArgs**, **FormatItemEventArgs**, **SeriesEventArgs**, **PdfDocumentEndedEventArgs**, **FormatNodeEventArgs**, **DraggingRowColumnEventArgs**, **RowColumnChangedEventArgs**, **UnknownFunctionEventArgs**, **ProgressEventArgs**, **QueryLoadingDataEventArgs**

Base class for event arguments.

Properties

- empty

Properties

- STATIC **empty**

Provides a value to use with events that do not have event data.

Type
EventArgs

Globalize Class

File
`wijmo.js`
Module
wijmo

Class that implements formatting and parsing of numbers and Dates.

By default, **Globalize** uses the American English culture. To switch cultures, include the appropriate **wijmo.culture.*.js** file after the wijmo files.

Methods

- format
- formatDate
- formatNumber
- getFirstDayOfWeek
- getNumberDecimalSeparator
- parseDate
- parseFloat
- parseInt

Methods

STATIC format

```
format(value: any, format: string, trim?: boolean, truncate?: boolean): string
```

Formats a number or a date.

The format strings used with the **format** function are similar to the ones used by **Globalize.js** and by the .NET Globalization library. The tables below contains links that describe the formats available:

- Standard Numeric Format Strings (http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx)
- Standard Date and Time Format Strings (http://msdn.microsoft.com/en-us/library/az4se3k1(v=vs.110).aspx)
- Custom Date and Time Format Strings (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)

Parameters

- **value**: **any**
  Number or Date to format (all other types are converted to strings).

- **format**: **string**
  Format string to use when formatting numbers or dates.

- **trim**: **boolean** OPTIONAL
  Whether to remove trailing zeros from numeric results.

- **truncate**: **boolean** OPTIONAL
  Whether to truncate the numeric values rather than round them.

Returns
string

STATIC formatDate

```
formatDate(value: Date, format: string): string
```

Formats a date using the current culture.

The **format** parameter contains a .NET-style **Date format string** with the following additions:

- *Q, q* Calendar quarter.
- *U* Fiscal quarter (government).
- *u* Fiscal quarter (private sector).
- *EEEE, EEE, EE, E* Fiscal year (government).
- *eeee, eee, ee, e* Fiscal year (private sector).

For example:

```
var d = new Date(2015, 9, 1); // Oct 1, 2015
console.log(wijmo.Globalize.format(d, '"FY"EEEE"Q"U') + ' (US culture)');
> FY2016Q1 (US culture)
```

Parameters

- **value**: **Date**
  Number or Date to format.

- **format**: **string**
  .NET-style Date format string.

Returns
string

STATIC **formatNumber**

```
formatNumber(value: number, format: string, trim?: boolean, truncate?: boolean): string
```

Formats a number using the current culture.

The **formatNumber** method accepts most .NET-style **Standard Numeric Format Strings (http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx)**, except for the 'e' and 'x' formats (scientific notation and hexadecimal) which are not supported.

Numeric format strings take the form *Axxccss*, where:

- *A* is a single case-insensitive alphabetic character called the format specifier.
- *xx* is an optional integer called the precision specifier. The precision specifier affects the number of digits in the result.
- *cc* is an optional string used to override the currency symbol when formatting currency values. This is useful when formatting currency values for cultures different than the current default (for example, when formatting Euro or Yen values in applications that use the English culture).
- *ss* is an optional string used to scale the number. If provided, it must consist of commas. The number is divided by 1000 for each comma specified.

The following table describes the standard numeric format specifiers and displays sample output produced by each format specifier for the default culture.

n Number: `formatNumber(1234.5, 'n2') => '1,234.50'`
f Fixed-point: `formatNumber(1234.5, 'f2') => '1234.50'`
g General (no trailing zeros): `formatNumber(1234.5, 'g2') => '1234.5'`
d Decimal (integers): `formatNumber(-1234, 'd6') => '-001234'`
x Hexadecimal (integers): `formatNumber(1234, 'x6') => '0004d2'`
c Currency: `formatNumber(1234, 'c') => '$ 1,234.00'`
p Percent: `formatNumber(0.1234, 'p2') => '12.34 %'`

The scaling specifier is especially useful when charting large values. For example, the markup below creates a chart that plots population versus GDP. The raw data expresses the population is units and the GDP in millions. The scaling specified in the axes formats causes the chart to show population in millions and GDP in trillions:

```
<wj-flex-chart
  items-source="countriesGDP" binding-x="pop" chart-type="Scatter">
  <wj-flex-chart-series
    name="GDP" binding="gdp"></wj-flex-chart-series>
  <wj-flex-chart-axis
    wj-property="axisX" title="Population (millions)"
    format="n0,,">
  </wj-flex-chart-axis>
  <wj-flex-chart-axis
    wj-property="axisY" title="GDP (US$ trillions)"
    format="c0,,">
  </wj-flex-chart-axis>
</wj-flex-chart>
```

Parameters

- **value**: **number**

Number to format.

- **format**: **string**
  .NET-style standard numeric format string (e.g. 'n2', 'c4', 'p0', 'g2', 'd2').

- **trim**: **boolean** OPTIONAL
  Whether to remove trailing zeros from the result.

- **truncate**: **boolean** OPTIONAL
  Whether to truncate the value rather than round it.

Returns
string

STATIC getFirstDayOfWeek

```
getFirstDayOfWeek(): number
```

Gets the first day of the week according to the current culture.

The value returned is between zero (Sunday) and six (Saturday).

Returns
number

STATIC getNumberDecimalSeparator

```
getNumberDecimalSeparator(): string
```

Gets the symbol used as a decimal separator in numbers.

Returns
string

STATIC parseDate

```
parseDate(value: string, format: string): Date
```

Parses a string into a Date.

Two-digit years are converted to full years based on the value of the calendar's **twoDigitYearMax** property. By default, this is set to 2029, meaning two-digit values of 30 to 99 are parsed as 19**, and values from zero to 29 are parsed as 20**.

You can change this threshold by assigning a new value to the calendar. For example:

```
// get calendar
var cal = wijmo.culture.Globalize.calendar;

// default threshold is 2029, so "30" is parsed as 1930
cal.twoDigitYearMax = 2029;
var d1 = wijmo.Globalize.parseDate('30/12', 'yy/MM'); // dec 1930
// changing threshold to 2100, so all values are parsed as 20**
cal.twoDigitYearMax = 2100;
var d2 = wijmo.Globalize.parseDate('30/12', 'yy/MM'); // dec 2030
```

Parameters

- **value**: **string**
  String to convert to a Date.

- **format**: **string**
  Format string used to parse the date.

Returns
Date

STATIC parseFloat

```
parseFloat(value: string, format?: string): number
```

Parses a string into a floating point number.

Parameters

- **value**: **string**
  String to convert to a number.

- **format**: **string** OPTIONAL
  Format to use when parsing the number.

Returns
number

STATIC parseInt

```
parseInt(value: string, format?: string): number
```

Parses a string into an integer.

Parameters

- **value**: **string**
  String to convert to an integer.

- **format**: **string** OPTIONAL
  Format to use when parsing the number.

Returns
number

Point Class

File
`wijmo.js`
Module
wijmo

Class that represents a point (with x and y coordinates).

Constructor

- constructor

Properties

- x
- y

Methods

- clone
- equals

Constructor

constructor

```
constructor(x?: number, y?: number): Point
```

Initializes a new instance of the **`Point`** class.

Parameters

- **x**: **number** OPTIONAL
  X coordinate of the new Point.
- **y**: **number** OPTIONAL
  Y coordinate of the new Point.

Returns
Point

Properties

x

Gets or sets the x coordinate of this **`Point`**.

Type
number

y

Gets or sets the y coordinate of this **`Point`**.

Type
number

Methods

clone

```
clone(): Point
```

Creates a copy of this **`Point`**.

Returns
`Point`

equals

```
equals(pt: Point): boolean
```

Returns true if a **`Point`** has the same coordinates as this **`Point`**.

Parameters

- **pt**: **Point**
  `Point` to compare to this **`Point`**.

Returns
boolean

PrintDocument Class

File
`wijmo.js`
Module
wijmo

Class that enables the creation of custom documents for printing.

The **PrintDocument** class makes it easy to create documents for printing or exporting to PDF. Most browsers allow you to select the paper size, orientation, margins, and whether to include page headers and footers.

To use, instantiate a **PrintDocument**, add content using the **append** method, and finish by calling the **print** method.

For example:

```
// create the document
var doc = new wijmo.PrintDocument({
  title: 'PrintDocument Test'
});

// add some simple text
doc.append('<h1>Printing Example</h1>');
doc.append('<p>This document was created using the <b>PrintDocument</b> class.</p>');

// add some existing elements
doc.append(document.getElementById('gaugeControl'));

// print the document (or export it to PDF)
doc.print();
```

Constructor

- constructor

Properties

- copyCss
- title

Methods

- addCSS
- append
- print

Constructor

constructor

```
constructor(options?: any): PrintDocument
```

Initializes a new instance of the **PrintDocument** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the **PrintDocument**.

Returns
PrintDocument

Properties

copyCss

Gets or sets a value that determines whether the **PrintDocument** should include the CSS style sheets defined in the main document.

Type
boolean

title

Gets or sets the document title.

Setting this property to null causes the **PrintDocument** to use the title from the current document.

Type
string

Methods

addCSS

```
addCSS(href: string): void
```

Adds a CSS style sheet to the document.

Parameters

- **href**: **string**
  URL of the CSS file that should be added to the document.

Returns
void

append

```
append(child: any): void
```

Appends an HTML element or string to the document.

Parameters

- **child**: **any**
  HTML element or string to append to the document.

Returns
void

print

```
print(): void
```

Prints the document.

Returns
void

PropertyChangedEventArgs Class

File
`wijmo.js`
Module
wijmo
Base Class
EventArgs

Provides arguments for property change events.

Constructor

- constructor

Properties

- empty
- newValue
- oldValue
- propertyName

Constructor

constructor

```
constructor(propertyName: string, oldValue: any, newValue: any): PropertyChangedEventArgs
```

Initializes a new instance of the **PropertyChangedEventArgs** class.

Parameters

- **propertyName**: **string**
  The name of the property whose value changed.
- **oldValue**: **any**
  The old value of the property.
- **newValue**: **any**
  The new value of the property.

Returns
PropertyChangedEventArgs

Properties

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

newValue

Gets the new value of the property.

Type
any

oldValue

Gets the old value of the property.

Type
any

propertyName

Gets the name of the property whose value changed.

Type
string

Rect Class

File
`wijmo.js`
Module
wijmo

Class that represents a rectangle (with left, top, width, and height).

Constructor

- constructor

Properties

- bottom
- height
- left
- right
- top
- width

Methods

- clone
- contains
- equals
- fromBoundingRect
- inflate
- intersection
- union

Constructor

constructor

```
constructor(left: number, top: number, width: number, height: number): Rect
```

Initializes a new instance of the **Rect** class.

Parameters

- **left**: **number**
  Left coordinate of the new **Rect**.
- **top**: **number**
  Top coordinate of the new **Rect**.
- **width**: **number**
  Width of the new **Rect**.
- **height**: **number**
  Height of the new **Rect**.

Returns
Rect

Properties

bottom

Gets the bottom coordinate of this **Rect**.

Type
number

height

Gets or sets the height of this **Rect**.

Type
number

left

Gets or sets the left coordinate of this **Rect**.

Type
number

right

Gets the right coordinate of this **Rect**.

Type
number

top

Gets or sets the top coordinate of this **Rect**.

Type
number

width

Gets or sets the width of this **Rect**.

Type
number

Methods

clone

```
clone(): Rect
```

Creates a copy of this **Rect**.

Returns
Rect

contains

```
contains(pt: any): boolean
```

Determines whether the rectangle contains a given point or rectangle.

Parameters

- **pt**: **any**
  The **`Point`** or **`Rect`** to ckeck.

Returns
boolean

equals

```
equals(rc: Rect): boolean
```

Returns true if a **`Rect`** has the same coordinates and dimensions as this **`Rect`**.

Parameters

- **rc**: **Rect**
  `Rect` to compare to this **`Rect`**.

Returns
boolean

STATIC fromBoundingRect

```
fromBoundingRect(rc: any): Rect
```

Creates a **`Rect`** from **ClientRect** or **SVGRect** objects.

Parameters

- **rc**: **any**
  Rectangle obtained by a call to the DOM's **getBoundingClientRect** or **GetBoundingBox** methods.

Returns
`Rect`

inflate

```
inflate(dx: number, dy: number): Rect
```

Creates a rectangle that results from expanding or shrinking a rectangle by the specified amounts.

Parameters

- **dx**: **number**
  The amount by which to expand or shrink the left and right sides of the rectangle.

- **dy**: **number**
  The amount by which to expand or shrink the top and bottom sides of the rectangle.

Returns
`Rect`

STATIC intersection

```
intersection(rc1: Rect, rc2: Rect): Rect
```

Gets a rectangle that represents the intersection of two rectangles.

Parameters

- **rc1**: **Rect**
  First rectangle.

- **rc2**: **Rect**
  Second rectangle.

Returns
`Rect`

STATIC union

```
union(rc1: Rect, rc2: Rect): Rect
```

Gets a rectangle that represents the union of two rectangles.

Parameters

- **rc1**: **Rect**
  First rectangle.

- **rc2**: **Rect**
  Second rectangle.

Returns
`Rect`

RequestErrorEventArgs Class

File
`wijmo.js`
Module
wijmo
Base Class
CancelEventArgs

Provides arguments for **XMLHttpRequest** error events.

Constructor

- constructor

Properties

- cancel
- empty
- message
- request

Constructor

constructor

```
constructor(xhr: XMLHttpRequest, msg?: string): RequestErrorEventArgs
```

Initializes a new instance of the **RequestErrorEventArgs** class.

Parameters

- **xhr**: **XMLHttpRequest**
  The **XMLHttpRequest** that detected the error. The status and statusText properties of the request object contain details about the error.

- **msg**: **string** OPTIONAL
  Optional error message.

Returns
RequestErrorEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

message

Gets or sets an error message to display to the user.

Type
string

request

Gets a reference to the **XMLHttpRequest** that detected the error.

The status and statusText properties of the request object contain details about the error.

Type
XMLHttpRequest

Size Class

File
`wijmo.js`
Module
wijmo

Class that represents a size (with width and height).

Constructor

- constructor

Properties

- height
- width

Methods

- clone
- equals

Constructor

constructor

```
constructor(width?: number, height?: number): Size
```

Initializes a new instance of the **Size** class.

Parameters

- **width**: **number** OPTIONAL
  Width of the new **Size**.
- **height**: **number** OPTIONAL
  Height of the new **Size**.

Returns
Size

Properties

height

Gets or sets the height of this **`Size`**.

Type
number

width

Gets or sets the width of this **`Size`**.

Type
number

Methods

clone

```
clone(): Size
```

Creates a copy of this **`Size`**.

Returns
`Size`

equals

```
equals(sz: Size): boolean
```

Returns true if a **`Size`** has the same dimensions as this **`Size`**.

Parameters

- **sz**: **Size**
  `Size` to compare to this **`Size`**.

Returns
boolean

Tooltip Class

File
`wijmo.js`
Module
wijmo
Derived Classes
ChartTooltip

Provides a pop-up window that displays additional information about elements on the page.

The **Tooltip** class can be used in two modes:

Automatic Mode: Use the **setTooltip** method to connect the **Tooltip** to one or more elements on the page. The **Tooltip** will automatically monitor events and display the tooltips when the user performs actions that trigger the tooltip. For example:

```
var tt = new wijmo.Tooltip();
tt.setTooltip('#menu', 'Select commands.');
tt.setTooltip('#tree', 'Explore the hierarchy.');
tt.setTooltip('#chart', '#idChartTooltip');
```

Manual Mode: The caller is responsible for showing and hiding the tooltip using the **show** and **hide** methods. For example:

```
var tt = new wijmo.Tooltip();
element.addEventListener('click', function () {
  if (tt.isVisible) {
    tt.hide();
  } else {
    tt.show(element, 'This is an important element!');
  }
});
```

Constructor

- constructor

Properties

- gap
- hideDelay
- isContentHtml
- isVisible
- showAtMouse
- showDelay

Methods

- dispose
- getTooltip
- hide
- onPopup
- setTooltip
- show

Events

- popup

Constructor

constructor

```
constructor(options?: any): Tooltip
```

Initializes a new instance of the **`Tooltip`** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the **`Tooltip`**.

Returns
Tooltip

Properties

gap

Gets or sets the distance between the tooltip and the target element.

Type
number

hideDelay

Gets or sets the delay, in milliseconds, before hiding the tooltip after the mouse leaves the target element.

Type
number

isContentHtml

Gets or sets a value that determines whether the tooltip contents should be displayed as plain text or as HTML.

Type
boolean

isVisible

Gets a value that determines whether the tooltip is currently visible.

Type
boolean

showAtMouse

Gets or sets a value that determines whether the tooltip should be positioned with respect to the mouse position rather than the target element.

Type
boolean

showDelay

Gets or sets the delay, in milliseconds, before showing the tooltip after the mouse enters the target element.

Type
number

Methods

dispose

```
dispose(): void
```

Removes all tooltips associated with this **`Tooltip`** instance.

Returns
void

getTooltip

```
getTooltip(element: any): string
```

Gets the tooltip content associated with a given element.

Parameters

- **element**: **any**
  Element, element ID, or control that the tooltip explains.

Returns
string

hide

```
hide(): void
```

Hides the tooltip if it is currently visible.

Returns
void

onPopup

```
onPopup(e: TooltipEventArgs): boolean
```

Raises the **popup** event.

Parameters

- **e**: **TooltipEventArgs**
  `TooltipEventArgs` that contains the event data.

Returns
boolean

setTooltip

```
setTooltip(element: any, content: string): void
```

Assigns tooltip content to a given element on the page.

The same tooltip may be used to display information for any number of elements on the page. To remove the tooltip from an element, call `setTooltip` and specify null for the content.

Parameters

- **element**: **any**
  Element, element ID, or control that the tooltip explains.

- **content**: **string**
  Tooltip content or ID of the element that contains the tooltip content.

Returns
void

show

```
show(element: any, content: string, bounds?: Rect): void
```

Shows the tooltip with the specified content, next to the specified element.

Parameters

- **element**: **any**
  Element, element ID, or control that the tooltip explains.
- **content**: **string**
  Tooltip content or ID of the element that contains the tooltip content.
- **bounds**: **Rect** OPTIONAL
  Optional element that defines the bounds of the area that the tooltip targets. If not provided, the bounds of the element are used (as reported by the **getBoundingClientRect** method).

Returns
void

Events

popup

Occurs before the tooltip content is displayed.

The event handler may customize the tooltip content or suppress the tooltip display by changing the event parameters.

Arguments
`TooltipEventArgs`

TooltipEventArgs Class

File
`wijmo.js`
Module
wijmo
Base Class
CancelEventArgs

Provides arguments for the **popup** event.

Constructor

- constructor

Properties

- cancel
- content
- empty

Constructor

constructor

```
constructor(content: string): TooltipEventArgs
```

Initializes a new instance of the **TooltipEventArgs** class.

Parameters

- **content**: **string**
  String to show in the tooltip.

Returns
TooltipEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

content

Gets or sets the content to show in the tooltip.

This parameter can be used while handling the **popup** event to modify the content of the tooltip.

Type
string

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

IEventHandler Interface

File
`wijmo.js`
Module
wijmo

Represents an event handler.

Event handlers are functions invoked when events are raised.

Every event handler has two arguments:

- **sender** is the object that raised the event, and
- **args** is an optional object that contains the event parameters.

IQueryInterface Interface

File
`wijmo.js`
Module
wijmo

Allows callers to verify whether an object implements an interface.

Methods

- implementsInterface

Methods

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the object implements a given interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Returns
boolean

Aggregate Enum

File
`wijmo.js`
Module
wijmo

Specifies the type of aggregate to calculate over a group of values.

Members

Name	Value	Description
None	0	No aggregate.
Sum	1	Returns the sum of the numeric values in the group.
Cnt	2	Returns the count of non-null values in the group.
Avg	3	Returns the average value of the numeric values in the group.
Max	4	Returns the maximum value in the group.
Min	5	Returns the minimum value in the group.
Rng	6	Returns the difference between the maximum and minimum numeric values in the group.
Std	7	Returns the sample standard deviation of the numeric values in the group (uses the formula based on n-1).
Var	8	Returns the sample variance of the numeric values in the group (uses the formula based on n-1).
StdPop	9	Returns the population standard deviation of the values in the group (uses the formula based on n).
VarPop	10	Returns the population variance of the values in the group (uses the formula based on n).
CntAll	11	Returns the count of all values in the group (including nulls).
First	12	Returns the first non-null value in the group.
Last	13	Returns the last non-null value in the group.

DataType Enum

File
`wijmo.js`
Module
wijmo

Specifies constants that represent data types.

Use the **getType** method to get a **DataType** from a value.

Members

Name	Value	Description
Object	0	Object (anything).
String	1	String.
Number	2	Number.
Boolean	3	Boolean.
Date	4	Date (date and time).
Array	5	Array.

Key Enum

File
`wijmo.js`
Module
wijmo

Specifies constants that represent keyboard codes.

This enumeration is useful when handling **keyDown** events.

Members

Name	Value	Description
Back	8	The backspace key.
Tab	9	The tab key.
Enter	13	The enter key.
Escape	27	The escape key.
Space	32	The space key.
PageUp	33	The page up key.
PageDown	34	The page down key.
End	35	The end key.
Home	36	The home key.
Left	37	The left arrow key.
Up	38	The up arrow key.
Right	39	The right arrow key.
Down	40	The down arrow key.
Delete	46	The delete key.
F1	112	The F1 key.
F2	113	The F2 key.
F3	114	The F3 key.
F4	115	The F4 key.
F5	116	The F5 key.
F6	117	The F6 key.
F7	118	The F7 key.
F8	119	The F8 key.
F9	120	The F9 key.
F10	121	The F10 key.
F11	122	The F11 key.
F12	123	The F12 key.

wijmo.collections Module

File
`wijmo.js`
Module
wijmo.collections

Defines interfaces and classes related to data, including the **ICollectionView** interface, **CollectionView** and **ObservableArray** classes.

Classes

- ArrayBase
- CollectionView
- CollectionViewGroup
- GroupDescription
- NotifyCollectionChangedEventArgs
- ObservableArray
- PageChangingEventArgs
- PropertyGroupDescription
- SortDescription

Interfaces

- ICollectionView
- IComparer
- IEditableCollectionView
- INotifyCollectionChanged
- IPagedCollectionView
- IPredicate

Enums

- NotifyCollectionChangedAction

ArrayBase Class

File
`wijmo.js`
Module
wijmo.collections
Derived Classes
ObservableArray

Base class for Array classes with notifications.

Constructor

- constructor

Constructor

constructor

```
constructor(): ArrayBase
```

Initializes a new instance of the **ArrayBase** class.

Returns
ArrayBase

CollectionView Class

File
`wijmo.js`
Module
wijmo.collections
Derived Classes
ODataCollectionView, **PivotCollectionView**
Implements
IEditableCollectionView, **IPagedCollectionView**

Class that implements the **ICollectionView** interface to expose data in regular JavaScript arrays.

The **CollectionView** class implements the following interfaces:

- **ICollectionView**: provides current record management, custom sorting, filtering, and grouping.
- **IEditableCollectionView**: provides methods for editing, adding, and removing items.
- **IPagedCollectionView**: provides paging.

To use the **CollectionView** class, start by declaring it and passing a regular array as a data source. Then configure the view using the **filter**, **sortDescriptions**, **groupDescriptions**, and **pageSize** properties. Finally, access the view using the **items** property. For example:

```
// create a new CollectionView
var cv = new wijmo.collections.CollectionView(myArray);

// sort items by amount in descending order
var sd = new wijmo.collections.SortDescription('amount', false);
cv.sortDescriptions.push(sd);

// show only items with amounts greater than 100
cv.filter = function(item) { return item.amount > 100 };

// show the sorted, filtered result on the console
for (var i = 0; i < cv.items.length; i++) {
  var item = cv.items[i];
  console.log(i + ': ' + item.name + ' ' + item.amount);
}
```

Constructor

- constructor

Properties

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- filter
- getError
- groupDescriptions
- groups
- isAddingNew
- isEditingItem
- isEmpty
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- newItemCreator
- pageCount
- pageIndex
- pageSize
- sortComparer
- sortConverter
- sortDescriptions
- sourceCollection
- totalItemCount
- trackChanges
- useStableSort

Methods

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh
- remove
- removeAt

Events

- collectionChanged
- currentChanged
- currentChanging
- pageChanged
- pageChanging
- sourceCollectionChanged

sourceCollectionChanging

Constructor

constructor

```
constructor(sourceCollection?: any, options?: any): CollectionView
```

Initializes a new instance of the **`CollectionView`** class.

Parameters

- **sourceCollection**: **any** OPTIONAL
  Array that serves as a source for this **`CollectionView`**.

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
CollectionView

Properties

canAddNew

Gets a value that indicates whether a new item can be added to the collection.

Type
boolean

canCancelEdit

Gets a value that indicates whether the collection view can discard pending changes and restore the original values of an edited object.

Type
boolean

canChangePage

Gets a value that indicates whether the **`pageIndex`** value can change.

Type
boolean

canFilter

Gets a value that indicates whether this view supports filtering via the **`filter`** property.

Type
boolean

canGroup

Gets a value that indicates whether this view supports grouping via the **`groupDescriptions`** property.

Type
boolean

canRemove

Gets a value that indicates whether items can be removed from the collection.

Type
boolean

canSort

Gets a value that indicates whether this view supports sorting via the **`sortDescriptions`** property.

Type
boolean

currentAddItem

Gets the item that is being added during the current add transaction.

Type
any

currentEditItem

Gets the item that is being edited during the current edit transaction.

Type
any

currentItem

Gets or sets the current item in the view.

Type
any

currentPosition

Gets the ordinal position of the current item in the view.

Type
number

filter

Gets or sets a callback used to determine if an item is suitable for inclusion in the view.

The callback function should return true if the item passed in as a parameter should be included in the view.

NOTE: If the filter function needs a scope (i.e. a meaningful 'this' value) remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
collectionView.filter = this._filter.bind(this);
```

Type
IPredicate

getError

Gets or sets a callback that determines whether a specific property of an item contains validation errors.

If provided, the callback should take two parameters containing the item and the property to validate, and should return a string describing the error (or null if there are no errors).

For example:

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? 'Cannot be negative!'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'No active items allowed in the US or UK!'
                    : null;
        }
        return null;
    }
});
```

Type
Function

groupDescriptions

Gets a collection of **GroupDescription** objects that describe how the items in the collection are grouped in the view.

Type
ObservableArray

groups

Gets an array of **CollectionViewGroup** objects that represents the top-level groups.

Type
CollectionViewGroup[]

isAddingNew

Gets a value that indicates whether an add transaction is in progress.

Type
boolean

isEditingItem

Gets a value that indicates whether an edit transaction is in progress.

Type
boolean

isEmpty

Gets a value that indicates whether this view contains no items.

Type
boolean

isPageChanging

Gets a value that indicates whether the page index is changing.

Type
boolean

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **`beginUpdate`** and **`endUpdate`**).

Type

itemCount

Gets the total number of items in the view taking paging into account.

Type
number

items

Gets items in the view.

Type
any[]

itemsAdded

Gets an **`ObservableArray`** containing the records that were added to the collection since **`trackChanges`** was enabled.

Type
`ObservableArray`

itemsEdited

Gets an **`ObservableArray`** containing the records that were edited in the collection since **`trackChanges`** was enabled.

Type
`ObservableArray`

itemsRemoved

Gets an **`ObservableArray`** containing the records that were removed from the collection since **`trackChanges`** was enabled.

Type
`ObservableArray`

newItemCreator

Gets or sets a function that creates new items for the collection.

If the creator function is not supplied, the **`CollectionView`** will try to create an uninitialized item of the appropriate type.

If the creator function is supplied, it should be a function that takes no parameters and returns an initialized object of the proper type for the collection.

Type
Function

pageCount

Gets the total number of pages.

Type
number

pageIndex

Gets the zero-based index of the current page.

Type
number

pageSize

Gets or sets the number of items to display on a page.

Type
number

sortComparer

Gets or sets a function used to compare values when sorting.

If provided, the sort comparer function should take as parameters two values of any type, and should return -1, 0, or +1 to indicate whether the first value is smaller than, equal to, or greater than the second. If the sort comparer returns null, the standard built-in comparer is used.

This **`sortComparer`** property allows you to use custom comparison algorithms that in some cases result in sorting sequences that are more consistent with user's expectations than plain string comparisons.

For example, see **Dave Koele's Alphanum algorithm**. It breaks up strings into chunks composed of strings or numbers, then sorts number chunks in value order and string chunks in ASCII order. Dave calls the result a "natural sorting order".

The example below shows a typical use for the **`sortComparer`** property:

```
// create a CollectionView with a custom sort comparer
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // custom comparer used for strings
      : null; // use default comparer used for everything else
  }
});
```

Type
Function

sortConverter

Gets or sets a function used to convert values when sorting.

If provided, the function should take as parameters a **SortDescription**, a data item, and a value to convert, and should return the converted value.

This property provides a way to customize sorting. For example, the **FlexGrid** control uses it to sort mapped columns by display value instead of by raw value.

For example, the code below causes a **CollectionView** to sort the 'country' property, which contains country code integers, using the corresponding country names:

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // convert country id into name
  }
  return value;
}
```

Type
Function

sortDescriptions

Gets a collection of **SortDescription** objects that describe how the items in the collection are sorted in the view.

Type
ObservableArray

sourceCollection

Gets or sets the underlying (unfiltered and unsorted) collection.

Type
any

totalItemCount

Gets the total number of items in the view before paging is applied.

Type
number

trackChanges

Gets or sets a value that determines whether the control should track changes to the data.

If **trackChanges** is set to true, the **CollectionView** keeps track of changes to the data and exposes them through the **itemsAdded**, **itemsRemoved**, and **itemsEdited** collections.

Tracking changes is useful in situations where you need to update the server after the user has confirmed that the modifications are valid.

After committing or cancelling changes, use the **clearChanges** method to clear the **itemsAdded**, **itemsRemoved**, and **itemsEdited** collections.

The **CollectionView** only tracks changes made when the proper **CollectionView** methods are used (**editItem**/**commitEdit**, **addNew**/**commitNew**, and **remove**). Changes made directly to the data are not tracked.

Type
boolean

useStableSort

Gets or sets whether to use a stable sort algorithm.

Stable sorting algorithms maintain the relative order of records with equal keys. For example, consider a collection of objects with an "Amount" field. If you sort the collection by "Amount", a stable sort will keep the original order of records with the same Amount value.

This property is false by default, which causes the **CollectionView** to use JavaScript's built-in sort method, which is very fast but not stable. Setting the **useStableSort** property to true increases sort times by 30% to 50%, which can be significant for large collections.

Type
boolean

Methods

addNew

```
addNew(): any
```

Creates a new item and adds it to the collection.

This method takes no parameters. It creates a new item, adds it to the collection, and defers refresh operations until the new item is committed using the **commitNew** method or canceled using the **cancelNew** method.

The code below shows how the **addNew** method is typically used:

```
// create the new item, add it to the collection
var newItem = view.addNew();

// initialize the new item
newItem.id = getFreshId();
newItem.name = 'New Customer';

// commit the new item so the view can be refreshed
view.commitNew();
```

You can also add new items by pushing them into the **sourceCollection** and then calling the **refresh** method. The main advantage of **addNew** is in user-interactive scenarios (like adding new items in a data grid), because it gives users the ability to cancel the add operation. It also prevents the new item from being sorted or filtered out of view until the add operation is committed.

Returns
any

beginUpdate

```
beginUpdate(): void
```

Suspend refreshes until the next call to **endUpdate**.

Returns
void

cancelEdit

```
cancelEdit(): void
```

Ends the current edit transaction and, if possible, restores the original value to the item.

Returns
void

cancelNew

```
cancelNew(): void
```

Ends the current add transaction and discards the pending new item.

Returns
void

clearChanges

```
clearChanges(): void
```

Clears all changes by removing all items in the **itemsAdded**, **itemsRemoved**, and **itemsEdited** collections.

Call this method after committing changes to the server or after refreshing the data from the server.

Returns
void

commitEdit

```
commitEdit(): void
```

Ends the current edit transaction and saves the pending changes.

Returns
void

commitNew

```
commitNew(): void
```

Ends the current add transaction and saves the pending new item.

Returns
void

contains

```
contains(item: any): boolean
```

Returns a value indicating whether a given item belongs to this view.

Parameters

- **item**: **any**
  Item to seek.

Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The collection will not be refreshed until the function finishes. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Returns
void

editItem

```
editItem(item: any): void
```

Begins an edit transaction of the specified item.

Parameters

- **item**: **any**
  Item to be edited.

Returns
void

endUpdate

```
endUpdate(): void
```

Resume refreshes suspended by a call to **beginUpdate**.

Returns
void

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

Calculates an aggregate value for the items in this collection.

Parameters

- **aggType**: **Aggregate**
  Type of aggregate to calculate.
- **binding**: **string**
  Property to aggregate on.
- **currentPage**: **boolean** OPTIONAL
  Whether to include only items on the current page.

Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Returns
boolean

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

Sets the specified item to be the current item in the view.

Parameters

- **item**: **any**
  Item that will become current.

Returns
boolean

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

Sets the first item in the view as the current item.

Returns
boolean

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

Sets the last item in the view as the current item.

Returns
boolean

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

Sets the item after the current item in the view as the current item.

Returns
boolean

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

Sets the item at the specified index in the view as the current item.

Parameters

- **index**: **number**
  Index of the item that will become current.

Returns
boolean

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

Sets the item before the current item in the view as the current item.

Returns
boolean

moveToFirstPage

```
moveToFirstPage(): boolean
```

Sets the first page as the current page.

Returns
boolean

moveToLastPage

```
moveToLastPage(): boolean
```

Sets the last page as the current page.

Returns
boolean

moveToNextPage

```
moveToNextPage(): boolean
```

Moves to the page after the current page.

Returns
boolean

moveToPage

```
moveToPage(index: number): boolean
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index of the page to move to.

Returns
boolean

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

Moves to the page before the current page.

Returns
boolean

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **`collectionChanged`** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Returns
void

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

Raises the **currentChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

Raises the **currentChanging** event.

Parameters

- **e**: **CancelEventArgs**
  `CancelEventArgs` that contains the event data.

Returns
boolean

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

Raises the **pageChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

Raises the **pageChanging** event.

Parameters

- **e**: **PageChangingEventArgs**
  PageChangingEventArgs that contains the event data.

Returns
boolean

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

Raises the **sourceCollectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

Raises the **sourceCollectionChanging** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Returns
boolean

refresh

```
refresh(): void
```

Re-creates the view using the current sort, filter, and group parameters.

Returns
void

remove

```
remove(item: any): void
```

Removes the specified item from the collection.

Parameters

- **item**: **any**
  Item to be removed from the collection.

Returns
void

removeAt

```
removeAt(index: number): void
```

Removes the item at the specified index from the collection.

Parameters

- **index**: **number**
  Index of the item to be removed from the collection. The index is relative to the view, not to the source collection.

Returns
void

Events

collectionChanged

Occurs when the collection changes.

Arguments
`NotifyCollectionChangedEventArgs`

currentChanged

Occurs after the current item changes.

Arguments
`EventArgs`

currentChanging

Occurs before the current item changes.

Arguments
`CancelEventArgs`

pageChanged

Occurs after the page index changes.

Arguments
`EventArgs`

pageChanging

Occurs before the page index changes.

Arguments
`PageChangingEventArgs`

sourceCollectionChanged

Occurs after the value of the **`sourceCollection`** property changes.

Arguments
`EventArgs`

sourceCollectionChanging

Occurs before the value of the **`sourceCollection`** property changes.

Arguments
`CancelEventArgs`

CollectionViewGroup Class

File
`wijmo.js`
Module
wijmo.collections

Represents a group created by a **CollectionView** object based on its **groupDescriptions** property.

Constructor

- constructor

Methods

- getAggregate

Constructor

constructor

```
constructor(groupDescription: GroupDescription, name: string, level: number, isBottomLevel: boolean): CollectionViewGroup
```

Initializes a new instance of the **CollectionViewGroup** class.

Parameters

- **groupDescription**: **GroupDescription**
  GroupDescription that owns the new group.
- **name**: **string**
  Name of the new group.
- **level**: **number**
  Level of the new group.
- **isBottomLevel**: **boolean**
  Whether this group has any subgroups.

Returns
CollectionViewGroup

Methods

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, view?: ICollectionView): void
```

Calculates an aggregate value for the items in this group.

Parameters

- **aggType**: **Aggregate**
  Type of aggregate to calculate.
- **binding**: **string**
  Property to aggregate on.
- **view**: **ICollectionView** OPTIONAL
  CollectionView that owns this group.

Returns
void

GroupDescription Class

File
`wijmo.js`
Module
wijmo.collections
Derived Classes
PropertyGroupDescription

Represents a base class for types defining grouping conditions.

The concrete class which is commonly used for this purpose is **PropertyGroupDescription**.

Methods

- groupNameFromItem
- namesMatch

Methods

- **groupNameFromItem**

```
groupNameFromItem(item: any, level: number): any
```

Returns the group name for the given item.

Parameters

- **item**: **any**
  The item to get group name for.
- **level**: **number**
  The zero-based group level index.

Returns
any

namesMatch

```
namesMatch(groupName: any, itemName: any): boolean
```

Returns a value that indicates whether the group name and the item name match (which implies that the item belongs to the group).

Parameters

- **groupName**: **any**
  The name of the group.

- **itemName**: **any**
  The name of the item.

Returns
boolean

NotifyCollectionChangedEventArgs Class

File
`wijmo.js`
Module
wijmo.collections
Base Class
EventArgs

Provides data for the **collectionChanged** event.

Constructor

- constructor

Properties

- action
- empty
- index
- item
- reset

Constructor

constructor

```
constructor(action?: NotifyCollectionChangedAction, item?, index?: number): NotifyCollectionChangedEventArgs
```

Initializes a new instance of the **NotifyCollectionChangedEventArgs** class.

Parameters

- **action**: **NotifyCollectionChangedAction** OPTIONAL
  Type of action that caused the event to fire.
- **item**: OPTIONAL
  Item that was added or changed.
- **index**: **number** OPTIONAL
  Index of the item.

Returns
NotifyCollectionChangedEventArgs

Properties

action

Gets the action that caused the event to fire.

Type
`NotifyCollectionChangedAction`

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

index

Gets the index at which the change occurred.

Type
number

item

Gets the item that was added, removed, or changed.

Type
any

STATIC reset

Provides a reset notification.

Type
`NotifyCollectionChangedEventArgs`

ObservableArray Class

File
`wijmo.js`
Module
wijmo.collections
Base Class
ArrayBase
Derived Classes
RowColCollection, **AxisCollection**, **PlotAreaCollection**, **SheetCollection**, **PivotFieldCollection**
Implements
INotifyCollectionChanged

Array that sends notifications on changes.

The class raises the **collectionChanged** event when changes are made with the push, pop, splice, insert, or remove methods.

Warning: Changes made by assigning values directly to array members or to the length of the array do not raise the **collectionChanged** event.

Constructor

- constructor

Properties

- isUpdating

Methods

- beginUpdate
- clear
- deferUpdate
- endUpdate
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(data?: any[]): ObservableArray
```

Initializes a new instance of the **ObservableArray** class.

Parameters

- **data**: **any**[] OPTIONAL
  Array containing items used to populate the **ObservableArray**.

Returns
ObservableArray

Properties

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **beginUpdate** and **endUpdate**).

Type

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Returns
void

clear

```
clear(): void
```

Removes all items from the array.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Returns
boolean

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

Searches for an item in the array.

Parameters

- **searchElement**: **any**
  Element to locate in the array.

- **fromIndex**: **number** OPTIONAL
  The index where the search should start.

Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **`collectionChanged`** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Returns
void

push

```
push(...item: any[]): number
```

Adds one or more items to the end of the array.

Parameters

- **...item**: **any**[]
  One or more items to add to the array.

Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes and/or adds items to the array.

Parameters

- **index**: **number**
  Position where items will be added or removed.
- **count**: **number**
  Number of items to remove from the array.
- **item**: **any** OPTIONAL
  Item to add to the array.

Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Arguments
NotifyCollectionChangedEventArgs

PageChangingEventArgs Class

File
wijmo.js
Module
wijmo.collections
Base Class
CancelEventArgs

Provides data for the **pageChanging** event

Constructor

- constructor

Properties

- cancel
- empty
- newPageIndex

Constructor

constructor

```
constructor(newIndex: number): PageChangingEventArgs
```

Initializes a new instance of the **PageChangingEventArgs** class.

Parameters
- **newIndex**: **number**
  Index of the page that is about to become current.

Returns
PageChangingEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

newPageIndex

Gets the index of the page that is about to become current.

Type
number

PropertyGroupDescription Class

File
wijmo.js
Module
wijmo.collections
Base Class
GroupDescription

Describes the grouping of items using a property name as the criterion.

For example, the code below causes a **CollectionView** to group items by the value of their 'country' property:

```
var cv = new wijmo.collections.CollectionView(items);
var gd = new wijmo.collections.PropertyGroupDescription('country');
cv.groupDescriptions.push(gd);
```

You may also specify a callback function that generates the group name. For example, the code below causes a **CollectionView** to group items by the first letter of the value of their 'country' property:

```
var cv = new wijmo.collections.CollectionView(items);
var gd = new wijmo.collections.PropertyGroupDescription('country',
  function(item, propName) {
    return item[propName][0]; // return country's initial
});
cv.groupDescriptions.push(gd);
```

Constructor

- constructor

Properties

- propertyName

Methods

- groupNameFromItem
- namesMatch

Constructor

constructor

```
constructor(property: string, converter?: Function): PropertyGroupDescription
```

Initializes a new instance of the **`PropertyGroupDescription`** class.

Parameters

- **property**: **string**
  The name of the property that specifies which group an item belongs to.

- **converter**: **Function** OPTIONAL
  A callback function that takes an item and a property name and returns the group name. If not specified, the group name is the property value for the item.

Returns
PropertyGroupDescription

Properties

propertyName

Gets the name of the property that is used to determine which group an item belongs to.

Type
string

Methods

groupNameFromItem

```
groupNameFromItem(item: any, level: number): any
```

Returns the group name for the given item.

Parameters

- **item**: **any**
  The item to get group name for.

- **level**: **number**
  The zero-based group level index.

Returns
any

namesMatch

```
namesMatch(groupName: any, itemName: any): boolean
```

Returns a value that indicates whether the group name and the item name match (which implies that the item belongs to the group).

Parameters

- **groupName**: **any**
  The name of the group.

- **itemName**: **any**
  The name of the item.

Returns
boolean

SortDescription Class

File
`wijmo.js`
Module
wijmo.collections

Describes a sorting criterion.

Constructor

- constructor

Properties

- ascending
- property
- property

Constructor

constructor

```
constructor(property: string, ascending: boolean): SortDescription
```

Initializes a new instance of the **`SortDescription`** class.

Parameters

- **property**: **string**
  Name of the property to sort on.

- **ascending**: **boolean**
  Whether to sort in ascending order.

Returns
SortDescription

Properties

ascending

Gets a value that determines whether to sort the values in ascending order.

Type
boolean

property

Gets the name of the property used to sort.

Type
string

ICollectionView Interface

File
`wijmo.js`
Module
wijmo.collections
Implements
INotifyCollectionChanged

Enables collections to have the functionalities of current record management, custom sorting, filtering, and grouping.

This is a JavaScript version of the **ICollectionView** interface used in Microsoft's XAML platform. It provides a consistent, powerful, and MVVM-friendly way to bind data to UI elements.

Wijmo includes several classes that implement **ICollectionView**. The most common is **CollectionView**, which works based on regular JavsScript arrays.

Properties

- canFilter
- canGroup
- canSort
- currentChanged
- currentChanging
- currentItem
- currentPosition
- filter
- groupDescriptions
- groups
- isEmpty
- items
- sortDescriptions
- sourceCollection

Methods

- beginUpdate
- contains
- deferUpdate
- endUpdate
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- refresh

Properties

canFilter

Gets a value that indicates whether this view supports filtering via the **`filter`** property.

Type
boolean

canGroup

Gets a value that indicates whether this view supports grouping via the **`groupDescriptions`** property.

Type
boolean

canSort

Gets a value that indicates whether this view supports sorting via the **`sortDescriptions`** property.

Type
boolean

currentChanged

Occurs after the current item changes.

Type
`Event`

currentChanging

Occurs before the current item changes.

Type
`Event`

currentItem

Gets the current item in the view.

Type
any

currentPosition

Gets the ordinal position of the current item in the view.

Type
number

filter

Gets or sets a callback used to determine if an item is suitable for inclusion in the view.

NOTE: If the filter function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
collectionView.filter = this._filter.bind(this);
```

Type
`IPredicate`

groupDescriptions

Gets a collection of **`GroupDescription`** objects that describe how the items in the collection are grouped in the view.

Type
`ObservableArray`

groups

Gets the top-level groups.

Type
any[]

isEmpty

Gets a value that indicates whether this view contains no items.

Type
boolean

items

Gets the filtered, sorted, grouped items in the view.

Type
any[]

sortDescriptions

Gets a collection of **`SortDescription`** objects that describe how the items in the collection are sorted in the view.

Type
`ObservableArray`

sourceCollection

Gets or sets the collection object from which to create this view.

Type
any

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends refreshes until the next call to **`endUpdate`**.

Returns
void

contains

```
contains(item: any): boolean
```

Returns a value that indicates whether a given item belongs to this view.

Parameters

- **item**: **any**
  The item to locate in the collection.

Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a beginUpdate/endUpdate block.

The collection will not be refreshed until the function has been executed. This method ensures endUpdate is called even if the function throws.

Parameters

- **fn**: **Function**
  Function to be executed within the beginUpdate/endUpdate block.

Returns
void

endUpdate

```
endUpdate(): void
```

Resumes refreshes suspended by a call to **beginUpdate**.

Returns
void

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

Sets the specified item to be the current item in the view.

Parameters

- **item**: **any**
  The item to set as the **`currentItem`**.

Returns
boolean

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

Sets the first item in the view as the current item.

Returns
boolean

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

Sets the last item in the view as the current item.

Returns
boolean

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

Sets the item after the current item in the view as the current item.

Returns
boolean

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

Sets the item at the specified index in the view as the current item.

Parameters

- **index**: **number**
  The index of the item to set as the **`currentItem`**.

Returns
boolean

moveCurrentToPrevious

```
moveCurrentToPrevious(): void
```

Sets the item before the current item in the view as the current item.

Returns
void

refresh

```
refresh(): void
```

Re-creates the view using the current sort, filter, and group parameters.

Returns
void

IComparer Interface

File
`wijmo.js`
Module
wijmo.collections

Represents the method that compares two objects.

IEditableCollectionView Interface

File
`wijmo.js`
Module
wijmo.collections
Implements
ICollectionView

Defines methods and properties that extend **ICollectionView** to provide editing capabilities.

Properties

- canAddNew
- canCancelEdit
- canRemove
- currentAddItem
- currentEditItem
- isAddingNew
- isEditingItem

Methods

- addNew
- cancelEdit
- cancelNew
- commitEdit
- commitNew
- editItem
- remove
- removeAt

Properties

canAddNew

Gets a value that indicates whether a new item can be added to the collection.

Type
boolean

canCancelEdit

Gets a value that indicates whether the collection view can discard pending changes and restore the original values of an edited object.

Type
boolean

canRemove

Gets a value that indicates whether items can be removed from the collection.

Type
boolean

currentAddItem

Gets the item that is being added during the current add transaction.

Type
any

currentEditItem

Gets the item that is being edited during the current edit transaction.

Type
any

isAddingNew

Gets a value that indicates whether an add transaction is in progress.

Type
boolean

isEditingItem

Gets a value that indicates whether an edit transaction is in progress.

Type
boolean

Methods

addNew

```
addNew(): any
```

Adds a new item to the collection.

Returns
any

cancelEdit

```
cancelEdit(): void
```

Ends the current edit transaction and, if possible, restores the original value to the item.

Returns
void

cancelNew

```
cancelNew(): void
```

Ends the current add transaction and discards the pending new item.

Returns
void

commitEdit

```
commitEdit(): void
```

Ends the current edit transaction and saves the pending changes.

Returns
void

commitNew

```
commitNew(): void
```

Ends the current add transaction and saves the pending new item.

Returns
void

editItem

```
editItem(item: any): void
```

Begins an edit transaction of the specified item.

Parameters

- **item**: **any**
  Item to edit.

Returns
void

remove

```
remove(item: any): void
```

Removes the specified item from the collection.

Parameters

- **item**: **any**
  Item to remove from the collection.

Returns
void

removeAt

```
removeAt(index: number): void
```

Removes the item at the specified index from the collection.

Parameters

- **index**: **number**
  Index of the item to remove from the collection.

Returns
void

INotifyCollectionChanged Interface

File
`wijmo.js`
Module
wijmo.collections

Notifies listeners of dynamic changes, such as when items get added and removed or when the collection is sorted, filtered, or grouped.

Properties

- collectionChanged

Properties

- **collectionChanged**

Occurs when the collection changes.

Type
Event

IPagedCollectionView Interface

File
`wijmo.js`
Module
wijmo.collections
Implements
ICollectionView

Defines methods and properties that extend **ICollectionView** to provide paging capabilities.

Properties

- canChangePage
- isPageChanging
- itemCount
- pageChanged
- pageChanging
- pageIndex
- pageSize
- totalItemCount

Methods

- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage

Properties

canChangePage

Gets a value that indicates whether the **`pageIndex`** value can change.

Type
boolean

isPageChanging

Gets a value that indicates whether the index is changing.

Type
boolean

itemCount

Gets the number of items in the view taking paging into account.

To get the total number of items, use the **`totalItemCount`** property.

Notice that this is different from the .NET **IPagedCollectionView**, where **itemCount** and **totalItemCount** both return the count before paging is applied.

Type
number

pageChanged

Occurs after the page index changes.

Type
`Event`

pageChanging

Occurs before the page index changes.

Type
`Event`

pageIndex

Gets the zero-based index of the current page.

Type
number

pageSize

Gets or sets the number of items to display on a page.

Type
number

totalItemCount

Gets the total number of items in the view before paging is applied.

To get the number of items in the current view not taking paging into account, use the **`itemCount`** property.

Notice that this is different from the .NET **IPagedCollectionView**, where **itemCount** and **totalItemCount** both return the count before paging is applied.

Type
number

Methods

moveToFirstPage

```
moveToFirstPage(): boolean
```

Sets the first page as the current page.

Returns
boolean

moveToLastPage

```
moveToLastPage(): boolean
```

Sets the last page as the current page.

Returns
boolean

moveToNextPage

```
moveToNextPage(): boolean
```

Moves to the page after the current page.

Returns
boolean

moveToPage

```
moveToPage(index: number): boolean
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index of the page to move to.

Returns
boolean

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

Moves to the page before the current page.

Returns
boolean

IPredicate Interface

File
`wijmo.js`
Module
wijmo.collections

Represents a method that takes an item of any type and returns a boolean that indicates whether the object meets a set of criteria.

NotifyCollectionChangedAction Enum

File
`wijmo.js`
Module
wijmo.collections

Describes the action that caused the **collectionChanged** event to fire.

Members

Name	Value	Description
Add	0	An item was added to the collection.
Remove	1	An item was removed from the collection.
Change	2	An item was changed or replaced.
Reset	3	Several items changed simultaneously (for example, the collection was sorted, filtered, or grouped).

wijmo.grid Module

File
`wijmo.grid.js`
Module
wijmo.grid

Defines the **FlexGrid** control and associated classes.

The example below creates a **FlexGrid** control and binds it to a 'data' array. The grid has four columns, specified by explicitly populating the grid's **columns** array.

Example

Show me (http://jsfiddle.net/Wijmo5/6GB66)

Classes

- CellEditEndingEventArgs
- CellFactory
- CellRange
- CellRangeEventArgs
- Column
- ColumnCollection
- DataMap
- FlexGrid
- FormatItemEventArgs
- GridPanel
- GroupRow
- HitTestInfo
- MergeManager
- Row
- RowCol
- RowColCollection
- RowCollection

Enums

- AllowDragging
- AllowMerging
- AllowResizing
- AutoSizeMode
- CellType
- HeadersVisibility
- KeyAction
- RowColFlags
- SelectedState
- SelectionMode
- SelMove

CellEditEndingEventArgs Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
CellRangeEventArgs

Provides arguments for the **cellEditEnding** event.

Constructor

- constructor

Properties

- cancel
- col
- data
- empty
- panel
- range
- row
- stayInEditMode

Constructor

constructor

```
constructor(p: GridPanel, rng: CellRange, data?: any): CellRangeEventArgs
```

Initializes a new instance of the **CellRangeEventArgs** class.

Parameters

- **p**: **GridPanel**
  GridPanel that contains the range.
- **rng**: **CellRange**
  Range of cells affected by the event.
- **data**: **any** OPTIONAL
  Data related to the event.

Inherited From
CellRangeEventArgs
Returns
CellRangeEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
CancelEventArgs
Type
boolean

col

Gets the column affected by this event.

Inherited From
CellRangeEventArgs
Type
number

data

Gets or sets the data associated with the event.

Inherited From
`CellRangeEventArgs`
Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

panel

Gets the **GridPanel** affected by this event.

Inherited From
`CellRangeEventArgs`
Type
`GridPanel`

range

Gets the `CellRange` affected by this event.

Inherited From
`CellRangeEventArgs`
Type
`CellRange`

row

Gets the row affected by this event.

Inherited From
`CellRangeEventArgs`
Type
number

stayInEditMode

Gets or sets whether the cell should remain in edit mode instead of finishing the edits.

Type
boolean

CellFactory Class

File
`wijmo.grid.js`
Module
wijmo.grid

Creates HTML elements that represent cells within a **FlexGrid** control.

Methods

- disposeCell
- getEditorValue
- updateCell

Methods

disposeCell

```
disposeCell(cell: HTMLElement): void
```

Disposes of a cell element and releases all resources associated with it.

Parameters

- **cell**: **HTMLElement**
  The element that represents the cell.

Returns
void

getEditorValue

```
getEditorValue(g: FlexGrid): any
```

Gets the value of the editor currently being used.

Parameters

- **g**: **FlexGrid**
  FlexGrid that owns the editor.

Returns
any

updateCell

```
updateCell(p: GridPanel, r: number, c: number, cell: HTMLElement, rng?: CellRange, updateContent?: boolean): void
```

Creates or updates a cell in the grid.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the cell.
- **r**: **number**
  The index of the row that contains the cell.
- **c**: **number**
  The index of the column that contains the cell.
- **cell**: **HTMLElement**
  The element that represents the cell.
- **rng**: **CellRange** OPTIONAL
  The **CellRange** object that contains the cell's merged range, or null if the cell is not merged.
- **updateContent**: **boolean** OPTIONAL
  Whether to update the cell's content as well as its position and style.

Returns
void

CellRange Class

File
`wijmo.grid.js`
Module
wijmo.grid

Represents a rectangular group of cells defined by two row indices and two column indices.

Constructor

- constructor

Properties

- bottomRow
- col
- col2
- columnSpan
- isSingleCell
- isValid
- leftCol
- rightCol
- row
- row2
- rowSpan
- topRow

Methods

- clone
- contains
- containsColumn
- containsRow
- equals
- getRenderSize
- intersects
- intersectsColumn
- intersectsRow
- setRange

Constructor

constructor

```
constructor(r?: number, c?: number, r2?: number, c2?: number): CellRange
```

Initializes a new instance of the **CellRange** class.

Parameters

- **r**: **number** OPTIONAL
  The index of the first row in the range (defaults to -1).
- **c**: **number** OPTIONAL
  The index of the first column in the range (defaults to -1).
- **r2**: **number** OPTIONAL
  The index of the last row in the range (defaults to **r**).
- **c2**: **number** OPTIONAL
  The index of the last column in the range (defaults to **c**).

Returns
CellRange

Properties

bottomRow

Gets the index of the bottom row in the range.

Type
number

col

Gets or sets the index of the first column in the range.

Type
number

col2

Gets or sets the index of the second column in the range.

Type
number

columnSpan

Gets the number of columns in the range.

Type
number

isSingleCell

Checks whether this range corresponds to a single cell (beginning and ending rows have the same index, and beginning and ending columns have the same index).

Type
boolean

isValid

Checks whether the range contains valid row and column indices (row and column values are zero or greater).

Type
boolean

leftCol

Gets the index of the leftmost column in the range.

Type
number

- **rightCol**

Gets the index of the rightmost column in the range.

Type
number

- **row**

Gets or sets the index of the first row in the range.

Type
number

- **row2**

Gets or sets the index of the second row in the range.

Type
number

- **rowSpan**

Gets the number of rows in the range.

Type
number

- **topRow**

Gets the index of the top row in the range.

Type
number

Methods

clone

```
clone(): CellRange
```

Creates a copy of the range.

Returns
CellRange

contains

```
contains(r: any, c?: number): boolean
```

Checks whether the range contains another range or a specific cell.

Parameters

- **r**: **any**
  The CellRange object or row index to find.

- **c**: **number** OPTIONAL
  The column index (required if the r parameter is not a CellRange object).

Returns
boolean

containsColumn

```
containsColumn(c: number): boolean
```

Checks whether the range contains a given column.

Parameters

- **c**: **number**
  The index of the column to find.

Returns
boolean

containsRow

```
containsRow(r: number): boolean
```

Checks whether the range contains a given row.

Parameters

- **r**: **number**
  The index of the row to find.

Returns
boolean

equals

```
equals(rng: CellRange): boolean
```

Checks whether the range equals another range.

Parameters

- **rng**: **CellRange**
  The CellRange object to compare to this range.

Returns
boolean

getRenderSize

```
getRenderSize(p: GridPanel): Size
```

Gets the rendered size of this range.

Parameters

- **p**: **GridPanel**
  The **GridPanel** object that contains the range.

Returns
Size

intersects

```
intersects(rng: CellRange): boolean
```

Checks whether the range intersects another range.

Parameters

- **rng**: **CellRange**
  The CellRange object to check.

Returns
boolean

intersectsColumn

```
intersectsColumn(rng: CellRange): boolean
```

Checks whether the range intersects the columns in another range.

Parameters

- **rng**: **CellRange**
  The CellRange object to check.

Returns
boolean

intersectsRow

```
intersectsRow(rng: CellRange): boolean
```

Checks whether the range intersects the rows in another range.

Parameters

- **rng**: **CellRange**
  The CellRange object to check.

Returns
boolean

setRange

```
setRange(r?: number, c?: number, r2?: number, c2?: number): void
```

Initializes an existing **CellRange**.

Parameters

- **r**: **number** OPTIONAL
  The index of the first row in the range (defaults to -1).

- **c**: **number** OPTIONAL
  The index of the first column in the range (defaults to -1).

- **r2**: **number** OPTIONAL
  The index of the last row in the range (defaults to **r**).

- **c2**: **number** OPTIONAL
  The index of the last column in the range (defaults to **c**).

Returns
void

CellRangeEventArgs Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
CancelEventArgs
Derived Classes
CellEditEndingEventArgs, **FormatItemEventArgs**, **XlsxFormatItemEventArgs**, **PdfFormatItemEventArgs**

Provides arguments for **CellRange** events.

Constructor

- constructor

Properties

- cancel
- col
- data
- empty
- panel
- range
- row

Constructor

constructor

```
constructor(p: GridPanel, rng: CellRange, data?: any): CellRangeEventArgs
```

Initializes a new instance of the **CellRangeEventArgs** class.

Parameters

- **p**: **GridPanel**
  GridPanel that contains the range.
- **rng**: **CellRange**
  Range of cells affected by the event.
- **data**: **any** OPTIONAL
  Data related to the event.

Returns
CellRangeEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
CancelEventArgs
Type
boolean

col

Gets the column affected by this event.

Type
number

data

Gets or sets the data associated with the event.

Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

panel

Gets the **GridPanel** affected by this event.

Type
GridPanel

range

Gets the **CellRange** affected by this event.

Type
CellRange

row

Gets the row affected by this event.

Type
number

Column Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
RowCol
Derived Classes
WjFlexGridColumn

Represents a column on the grid.

Constructor

- constructor

Properties

- aggregate
- align
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- binding
- collectionView
- cssClass
- currentSort
- dataMap
- dataType
- dropDownCssClass
- format
- grid
- header
- index
- inputType
- isContentHtml
- isReadOnly
- isRequired
- isSelected
- isVisible
- mask
- maxLength
- maxWidth
- minWidth
- name
- pos
- quickAutoSize
- renderSize
- renderWidth
- showDropDown
- size
- sortMemberPath
- visible
- visibleIndex
- width
- wordWrap

Methods

- getAlignment
- getIsRequired
- onPropertyChanged

Constructor

constructor

```
constructor(options?: any): Column
```

Initializes a new instance of the **Column** class.

Parameters

- **options**: **any** OPTIONAL
  Initialization options for the column.

Returns
Column

Properties

aggregate

Gets or sets the **Aggregate** to display in the group header rows for the column.

Type
Aggregate

align

Gets or sets the horizontal alignment of items in the column.

The default value for this property is null, which causes the grid to select the alignment automatically based on the column's **dataType** (numbers are right-aligned, Boolean values are centered, and other types are left-aligned).

If you want to override the default alignment, set this property to 'left', 'right', or 'center'.

Type
string

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Inherited From
RowCol
Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Inherited From
`RowCol`
Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Inherited From
`RowCol`
Type
boolean

allowSorting

Gets or sets a value that indicates whether the user can sort the column by clicking its header.

Type
boolean

binding

Gets or sets the name of the property the column is bound to.

Type
string

collectionView

Gets the `ICollectionView` bound to this row or column.

Inherited From
`RowCol`
Type
`ICollectionView`

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Inherited From
`RowCol`
Type
string

currentSort

Gets a string that describes the current sorting applied to the column. Possible values are '+' for ascending order, '-' for descending order, or null for unsorted columns.

Type
string

dataMap

Gets or sets the **`DataMap`** used to convert raw values into display values for the column.

Columns with an associated **`dataMap`** show drop-down buttons that can be used for quick editing. If you do not want to show the drop-down buttons, set the column's **`showDropDown`** property to false.

Cell drop-downs require the wijmo.input module to be loaded.

Type
`DataMap`

dataType

Gets or sets the type of value stored in the column.

Values are coerced into the proper type when editing the grid.

Type
`DataType`

dropDownCssClass

Gets or sets a CSS class name to add to drop-downs in this column.

The drop-down buttons are shown only if the column has a **`dataMap`** set and is editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Type
string

format

Gets or sets the format string used to convert raw values into display values for the column (see **`Globalize`**).

Type
string

grid

Gets the **`FlexGrid`** that owns the row or column.

Inherited From
`RowCol`
Type
`FlexGrid`

header

Gets or sets the text displayed in the column header.

Type
string

index

Gets the index of the row or column in the parent collection.

Inherited From
`RowCol`
Type
number

inputType

Gets or sets the "type" attribute of the HTML input element used to edit values in this column.

By default, this property is set to "tel" for numeric columns, and to "text" for all other non-boolean column types. The "tel" input type causes mobile devices to show a numeric keyboard that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In these cases, try setting the property to "number" or simply "text."

Type
string

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Inherited From
`RowCol`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Inherited From
`RowCol`
Type
boolean

isRequired

Gets or sets a value that determines whether values in the column are required.

By default, this property is set to null, which means values are required, but non-masked string columns may contain empty strings.

When set to true, values are required and empty strings are not allowed.

When set to false, null values and empty strings are allowed.

Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Inherited From
`RowCol`
Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Inherited From
`RowCol`
Type
boolean

mask

Gets or sets a mask to use while editing values in this column.

The mask format is the same used by the **`InputMask`** control.

If specified, the mask must be compatible with the value of the **`format`** property. For example, the mask '99/99/9999' can be used for entering dates formatted as 'MM/dd/yyyy'.

Type
string

maxLength

Gets or sets the maximum number of characters that the can be entered into the cell.

Set this property to null to allow entry of any number of characters.

Type
number

maxWidth

Gets or sets the maximum width of the column.

Type
number

minWidth

Gets or sets the minimum width of the column.

Type
number

name

Gets or sets the name of the column.

The column name can be used to retrieve the column using the **`getColumn`** method.

Type
string

pos

Gets the position of the row or column.

Inherited From
`RowCol`
Type
number

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing this column.

Setting this property to false disables quick auto-sizing for this column. Setting it to true enables the feature, subject to the value of the grid's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for columns that display plain text and don't have templates.

Type
boolean

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Inherited From
`RowCol`
Type
number

renderWidth

Gets the render width of the column.

The value returned takes into account the column's visibility, default size, and min and max sizes.

Type
number

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in this column.

The drop-down buttons are shown only if the column has a **`dataMap`** set and is editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Type
boolean

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`RowCol`
Type
number

sortMemberPath

Gets or sets the name of the property to use when sorting this column.

Use this property in cases where you want the sorting to be performed based on values other than the ones specified by the **`binding`** property.

Setting this property is null causes the grid to use the value of the **`binding`** property to sort the column.

Type
string

visible

Gets or sets a value that indicates whether the row or column is visible.

Inherited From
`RowCol`
Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Inherited From
`RowCol`
Type
number

width

Gets or sets the width of the column.

Column widths may be positive numbers (sets the column width in pixels), null or negative numbers (uses the collection's default column width), or strings in the format '{number}*' (star sizing).

The star-sizing option performs a XAML-style dynamic sizing where column widths are proportional to the number before the star. For example, if a grid has three columns with widths "100", "*", and "3*", the first column will be 100 pixels wide, the second will take up 1/4th of the remaining space, and the last will take up the remaining 3/4ths of the remaining space.

Star-sizing allows you to define columns that automatically stretch to fill the width available. For example, set the width of the last column to "*" and it will automatically extend to fill the entire grid width so there's no empty space. You may also want to set the column's **`minWidth`** property to prevent the column from getting too narrow.

Type
any

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Inherited From
`RowCol`
Type
boolean

Methods

getAlignment

```
getAlignment(): string
```

Gets the actual column alignment.

Returns the value of the **`align`** property if it is not null, or selects the alignment based on the column's **`dataType`**.

Returns
string

getIsRequired

```
getIsRequired(): boolean
```

Gets a value that determines whether the column is required.

Returns the value of the **`isRequired`** property if it is not null, or determines the required status based on the column's **`dataType`**.

By default, string columns are not required unless they have an associated **`dataMap`** or **`mask`**; all other data types are required.

Returns
boolean

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Inherited From
`RowCol`
Returns
void

ColumnCollection Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
RowColCollection

Represents a collection of **Column** objects in a **FlexGrid** control.

Constructor

- constructor

Properties

- defaultSize
- firstVisibleIndex
- frozen
- isUpdating
- maxSize
- minSize
- visibleLength

Methods

- beginUpdate
- canMoveElement
- clear
- deferUpdate
- endUpdate
- getColumn
- getItemAt
- getNextCell
- getTotalSize
- implementsInterface
- indexOf
- insert
- isFrozen
- moveElement
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(g: FlexGrid, defaultSize: number): RowColCollection
```

Initializes a new instance of the **RowColCollection** class.

Parameters

- **g**: **FlexGrid**
  The **FlexGrid** that owns the collection.

- **defaultSize**: **number**
  The default size of the elements in the collection.

Inherited From
RowColCollection
Returns
RowColCollection

Properties

defaultSize

Gets or sets the default size of elements in the collection.

Inherited From
RowColCollection
Type
number

firstVisibleIndex

Gets the index of the first visible column (where the outline tree is displayed).

Type

frozen

Gets or sets the number of frozen rows or columns in the collection.

Frozen rows and columns do not scroll, and instead remain at the top or left of the grid, next to the fixed cells. Unlike fixed cells, however, frozen cells may be selected and edited like regular cells.

Inherited From
`RowColCollection`
Type
number

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **`beginUpdate`** and **`endUpdate`**).

Inherited From
`ObservableArray`
Type

maxSize

Gets or sets the maximum size of elements in the collection.

Inherited From
`RowColCollection`
Type
number

minSize

Gets or sets the minimum size of elements in the collection.

Inherited From
`RowColCollection`
Type
number

visibleLength

Gets the number of visible elements in the collection (**`isVisible`**).

Inherited From
`RowColCollection`
Type
number

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`RowColCollection`
Returns
void

canMoveElement

```
canMoveElement(src: number, dst: number): boolean
```

Checks whether an element can be moved from one position to another.

Parameters

- **src**: **number**
  The index of the element to move.

- **dst**: **number**
  The position to which to move the element, or specify -1 to append the element.

Inherited From
`RowColCollection`
Returns
boolean

clear

```
clear(): void
```

Removes all items from the array.

Inherited From
ObservableArray
Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Inherited From
ObservableArray
Returns
void

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Returns
Column

getItemAt

```
getItemAt(position: number): number
```

Gets the index of the element at a given physical position.

Parameters

- **position**: **number**
  Position of the item in the collection, in pixels.

Inherited From
RowColCollection
Returns
number

getNextCell

```
getNextCell(index: number, move: SelMove, pageSize: number): void
```

Finds the next visible cell for a selection change.

Parameters

- **index**: **number**
  Starting index for the search.
- **move**: **SelMove**
  Type of move (size and direction).
- **pageSize**: **number**
  Size of a page (in case the move is a page up/down).

Inherited From
`RowColCollection`
Returns
void

getTotalSize

```
getTotalSize(): number
```

Gets the total size of the elements in the collection.

Inherited From
`RowColCollection`
Returns
number

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
ObservableArray
Returns
boolean

indexOf

```
indexOf(name: any): number
```

Gets the index of a column by name or binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **any**
  The name or binding to find.

Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Inherited From
`ObservableArray`
Returns
void

isFrozen

```
isFrozen(index: number): boolean
```

Checks whether a row or column is frozen.

Parameters

- **index**: **number**
  The index of the row or column to check.

Inherited From
`RowColCollection`
Returns
boolean

moveElement

```
moveElement(src: number, dst: number): void
```

Moves an element from one position to another.

Parameters

- **src**: **number**
  Index of the element to move.

- **dst**: **number**
  Position where the element should be moved to (-1 to append).

Inherited From
`RowColCollection`
Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Keeps track of dirty state and invalidate grid on changes.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL

Inherited From
`RowColCollection`
Returns
void

push

```
push(item: any): number
```

Appends an item to the array.

Parameters

- **item**: **any**
  Item to add to the array.

Inherited From
`RowColCollection`
Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
`ObservableArray`
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Inherited From
ObservableArray
Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
ObservableArray
Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
ObservableArray
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes or adds items to the array.

Parameters

- **index**: **number**
  Position where items are added or removed.
- **count**: **number**
  Number of items to remove from the array.
- **item**: **any** OPTIONAL
  Item to add to the array.

Inherited From
`RowColCollection`
Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

DataMap Class

File
`wijmo.grid.js`
Module
wijmo.grid

Represents a data map for use with a column's **dataMap** property.

Data maps provide the grid with automatic look up capabilities. For example, you may want to display a customer name instead of his ID, or a color name instead of its RGB value.

The code below binds a grid to a collection of products, then assigns a **DataMap** to the grid's 'CategoryID' column so the grid displays the category names rather than the raw IDs.

The grid takes advantage of data maps also for editing. If the **wijmo.input** module is loaded, then when editing data-mapped columns the grid will show a drop-down list containing the values on the map.

```
// bind grid to products
var flex = new wijmo.grid.FlexGrid();
flex.itemsSource = products;

// map CategoryID column to show category name instead of ID
var col = flex.columns.getColumn('CategoryID');
col.dataMap = new wijmo.grid.DataMap(categories, 'CategoryID', 'CategoryName');
```

In general, data maps apply to whole columns. However, there are situations where you may want to restrict the options available for a cell based on a value on a different column. For example, if you have "Country" and "City" columns, you will probably want to restrict the cities based on the current country.

There are two ways you can implement these "dynamic" data maps:

1. If the **DataMap** is just a list of strings, you can change it before the grid enters edit mode. In this case, the cells contain the string being displayed, and changing the map won't affect other cells in the same column. This fiddle demonstrates: show me (http://jsfiddle.net/Wijmo5/8brL80r8/).
2. If the **DataMap** is a real map (stores key values in the cells, shows a corresponding string), then you can apply a filter to restrict the values shown in the drop-down. The **DataMap** will still contain the same keys and values, so other cells in the same column won't be disturbed by the filter. This fiddle demonstrates: show me (http://jsfiddle.net/Wijmo5/xborLd4t/).

In some cases, you may want to create a **DataMap** to represent an enumeration. This can be done with the following code:

```
// build a DataMap for a given enum
 function getDataMap(enumClass) {
     var pairs = [];
     for (var key in enumClass) {
         var val = parseInt(key);
         if (!isNaN(val)) {
             pairs.push({ key: val, name: enumClass[val] });
         }
     }
     return new wijmo.grid.DataMap(pairs, 'key', 'name');
}
```

Constructor

- constructor

Properties

- collectionView
- displayMemberPath
- isEditable
- selectedValuePath
- sortByDisplayValues

Methods

- getDisplayValue
- getDisplayValues
- getKeyValue
- getKeyValues
- onMapChanged

Events

- mapChanged

Constructor

constructor

```
constructor(itemsSource: any, selectedValuePath?: string, displayMemberPath?: string): DataMap
```

Initializes a new instance of the **DataMap** class.

Parameters

- **itemsSource**: **any**
  An array or **ICollectionView** that contains the items to map.
- **selectedValuePath**: **string** OPTIONAL
  The name of the property that contains the keys (data values).
- **displayMemberPath**: **string** OPTIONAL
  The name of the property to use as the visual representation of the items.

Returns
DataMap

Properties

collectionView

Gets the **`ICollectionView`** object that contains the map data.

Type
`ICollectionView`

displayMemberPath

Gets the name of the property to use as the visual representation of the item.

Type
string

isEditable

Gets or sets a value that indicates whether users should be allowed to enter values that are not present on the **`DataMap`**.

In order for a **`DataMap`** to be editable, the **`selectedValuePath`** and **`displayMemberPath`** must be set to the same value.

Type
boolean

selectedValuePath

Gets the name of the property to use as a key for the item (data value).

Type
string

sortByDisplayValues

Gets or sets a value that determines whether to use mapped (display) or raw values when sorting the data.

Type
boolean

Methods

getDisplayValue

```
getDisplayValue(key: any): any
```

Gets the display value that corresponds to a given key.

Parameters

- **key**: **any**
  The key of the item to retrieve.

Returns
any

getDisplayValues

```
getDisplayValues(dataItem?: any): string[]
```

Gets an array with all of the display values on the map.

Parameters

- **dataItem**: **any** OPTIONAL
  Data item for which to get the display items. This parameter is optional. If not provided, all possible display values should be returned.

Returns
string[]

getKeyValue

```
getKeyValue(displayValue: string): any
```

Gets the key that corresponds to a given display value.

Parameters

- **displayValue**: **string**
  The display value of the item to retrieve.

Returns
any

getKeyValues

```
getKeyValues(): string[]
```

Gets an array with all of the keys on the map.

Returns
string[]

onMapChanged

```
onMapChanged(e?: EventArgs): void
```

Raises the **mapChanged** event.

Parameters
- **e**: **EventArgs** OPTIONAL

Returns
void

Events

mapChanged

Occurs when the map data changes.

Arguments
EventArgs

FlexGrid Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
Control
Derived Classes
MultiRow, **FlexSheet**, **PivotGrid**, **WjFlexGrid**

The **FlexGrid** control provides a powerful and flexible way to display and edit data in a tabular format.

The **FlexGrid** control is a full-featured grid, providing all the features you are used to including several selection modes, sorting, column reordering, grouping, filtering, editing, custom cells, XAML-style star-sizing columns, row and column virtualization, etc.

The **FlexGrid** control supports the following keyboard commands:

Key Combination	**Action**
Shift + Space	Select row
Control + Space	Select column
F2	Start editing the current cell
Space	Start editing or toggle checkbox
Control + A	Select the entire grid contents
Left/Right	Select the cell to the left/right of the selection, collapse/expand group rows
Shift + Left/Right	Extend the selection to include the next cell to the left/right of the selection
Up/Down	Select the next cell above or below the selection
Shift + Up/Down	Extend the selection to include the cell above or below the selection
Alt + Up/Down	Drop down the listbox editor for the current cell
PageUp/Down	Select the cell one page above or below the selection
Shift + PageUp/Down	Extend the selection to include the cell one page above or below the selection
Alt + PageUp/Down	Move the selection to the first or last row
Shift + Alt + PageUp/Down	Extend the selection to include the first or last row
Home/End	Move the selection to the first or last column
Shift + Home/End	Extend the selection to include the first or last column
Ctrl + Home/End	Move the selection to the first/last row and column
Shift + Ctrl + Home/End	Extend the selection to include the first/last row and column
Escape	Cancel current cell or row editing operation
Tab	Move the selection to the next focusable element on the page (by default, can be overridden using the **keyActionTab** property)
Enter	Exit editing mode and move the selection to the cell below the current one (by default, can be overridden using the **keyActionEnter** property)
Delete, Backspace	Delete the currently selected rows (if the **allowDelete** property is set to true), or clear the content of the selected cells (if the values are not required).
Control + C or Control + Insert	Copy the selection to the clipboard (if the **autoClipboard** property is set to true)
Control + V or Shift + Insert	Paste the content of the clipboard into the selected area (if the **autoClipboard** property is set to true)

Example

Show me (http://jsfiddle.net/Wijmo5/6GB66)

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- deferResizing
- editableCollectionView
- editRange
- frozenColumns
- frozenRows
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- validateEdits
- viewRange
- virtualizationThreshold

Methods

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- deferUpdate
- dispose

- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- loadedRows
- loadingRows

- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): FlexGrid
```

Initializes a new instance of the **FlexGrid** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
FlexGrid

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **`isReadOnly`** property is set to true.

Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **itemsSource**.

The column generation depends on the **itemsSource** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **autoGenerateColumns** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Type
boolean

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Type
AutoSizeMode

bottomLeftCells

Gets the **GridPanel** that contains the bottom left cells.

The **bottomLeftCells** panel appears below the row headers, to the left of the **columnFooters** panel.

Type
GridPanel

cellFactory

Gets or sets the **`CellFactory`** that creates and updates cells for this grid.

Type
`CellFactory`

cells

Gets the **`GridPanel`** that contains the data cells.

Type
`GridPanel`

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Type
`Size`

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Type
boolean

collectionView

Gets the **ICollectionView** that contains the grid data.

Type
ICollectionView

columnFooters

Gets the **GridPanel** that contains the column footer cells.

The **columnFooters** panel appears below the grid cells, to the right of the **bottomLeftCells** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **columnFooters** panel to display summary data for columns that have the **aggregate** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Type
GridPanel

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Type
string

columns

Gets the grid's column collection.

Type
ColumnCollection

controlRect

Gets the bounding rectangle of the control in page coordinates.

Type
Rect

STATIC controlTemplate

Gets or sets the template used to instantiate **FlexGrid** controls.

Type
any

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **`deferResizing`** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Type
boolean

editableCollectionView

Gets the **`IEditableCollectionView`** that contains the grid data.

Type
`IEditableCollectionView`

editRange

Gets a **`CellRange`** that identifies the cell currently being edited.

Type
`CellRange`

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Type
number

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Type
`HeadersVisibility`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Type
Function

itemsSource

Gets or sets the array or **ICollectionView** that contains items shown on the grid.

Type
any

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **getError** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **MoveDown**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Type
KeyAction

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **None**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **Cycle**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **CycleOut** setting that causes the selection to move through the cells (as **Cycle**), and then on to the next/previous control on the page when the last or first cells are selected.

Type
KeyAction

mergeManager

Gets or sets the **MergeManager** object responsible for determining how cells should be merged.

Type
MergeManager

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **newRowAtTop** property to true, and you want the new row template to remain visible at all times, set the **frozenRows** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **allowAddNew** property is set to true and if the **itemsSource** object supports adding new items.

Type
boolean

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **preserveOutlineState** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Type
string

rowHeaders

Gets the **`GridPanel`** that contains the row header cells.

Type
`GridPanel`

rows

Gets the grid's row collection.

Type
`RowCollection`

scrollPosition

Gets or sets a `Point` that represents the value of the grid's scrollbars.

Type
`Point`

scrollSize

Gets the size of the grid content in pixels.

Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Type
any[]

selection

Gets or sets the current selection.

Type
`CellRange`

selectionMode

Gets or sets the current selection mode.

Type
`SelectionMode`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **`showDropDown`** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **`itemValidator`** property or the **`getError`** property on the grid's **`itemsSource`**.

Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Type
boolean

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Type
boolean

topLeftCells

Gets the **`GridPanel`** that contains the top left cells (to the left of the column headers).

Type
`GridPanel`

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Type
number

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **`getError`** method on the grid's **`itemsSource`**.

Type
boolean

viewRange

Gets the range of cells currently in view.

Type
`CellRange`

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.

Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
`Control`
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Returns
void

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Returns
`Column`

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.
- **c**: **number**
  Column index of the cell to inspect.

Returns
SelectedState

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Returns
boolean

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Returns
boolean

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Returns
boolean

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Returns
boolean

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the grid layout and content, or just the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.

- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.

- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.
- **c**: **number**
  Index of the column to scroll into view.

Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

Parameters

- **rng**: **any**
  Range to select.
- **show**: **any** OPTIONAL
  Whether to scroll the new selection into view.

Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Sets the value of a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the grid to show the change.

Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Hidden rows and columns are skipped.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.
- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Arguments
CellRangeEventArgs

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Arguments
CellRangeEventArgs

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Arguments
`CellEditEndingEventArgs`

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Arguments
`CellRangeEventArgs`

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

The event handler may cancel the copy operation.

Arguments
`CellRangeEventArgs`

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **`allowDelete`** property).

Arguments
`CellRangeEventArgs`

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **`allowDelete`** property).

The event handler may cancel the row deletion.

Arguments
`CellRangeEventArgs`

draggedColumn

Occurs when the user finishes dragging a column.

Arguments
`CellRangeEventArgs`

draggedRow

Occurs when the user finishes dragging a row.

Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Arguments
`CellRangeEventArgs`

draggingRowOver

Occurs as the user drags a row to a new position.

Arguments
`CellRangeEventArgs`

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **itemFormatter** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Arguments
FormatItemEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Arguments
CellRangeEventArgs

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Arguments
CellRangeEventArgs

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Arguments
`CellRangeEventArgs`

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Arguments
CellRangeEventArgs

resizedColumn

Occurs when the user finishes resizing a column.

Arguments
CellRangeEventArgs

resizedRow

Occurs when the user finishes resizing rows.

Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Arguments
`CellRangeEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Arguments
`CellRangeEventArgs`

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **`rowEditStarted`** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Arguments
`CellRangeEventArgs`

rowEditStarted

Occurs after a row enters edit mode.

Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Arguments
`EventArgs`

selectionChanged

Occurs after selection changes.

Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Arguments
`CellRangeEventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Arguments
`CellRangeEventArgs`

updatedLayout

Occurs after the grid has updated its internal layout.

Arguments
`EventArgs`

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Arguments
`EventArgs`

updatingLayout

Occurs before the grid updates its internal layout.

Arguments
`CancelEventArgs`

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Arguments
`CancelEventArgs`

FormatItemEventArgs Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
CellRangeEventArgs

Provides arguments for the **formatItem** event.

Constructor

- constructor

Properties

- cancel
- cell
- col
- data
- empty
- panel
- range
- row

Constructor

constructor

```
constructor(p: GridPanel, rng: CellRange, cell: HTMLElement): FormatItemEventArgs
```

Initializes a new instance of the **`FormatItemEventArgs`** class.

Parameters

- **p**: **GridPanel**
  `GridPanel` that contains the range.
- **rng**: **CellRange**
  Range of cells affected by the event.
- **cell**: **HTMLElement**
  Element that represents the grid cell to be formatted.

Returns
FormatItemEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

cell

Gets a reference to the element that represents the grid cell to be formatted.

Type
HTMLElement

col

Gets the column affected by this event.

Inherited From
`CellRangeEventArgs`
Type
number

data

Gets or sets the data associated with the event.

Inherited From
`CellRangeEventArgs`
Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

panel

Gets the **GridPanel** affected by this event.

Inherited From
`CellRangeEventArgs`
Type
`GridPanel`

range

Gets the `CellRange` affected by this event.

Inherited From
`CellRangeEventArgs`
Type
`CellRange`

row

Gets the row affected by this event.

Inherited From
`CellRangeEventArgs`
Type
number

GridPanel Class

File
`wijmo.grid.js`
Module
wijmo.grid
Derived Classes
FlexSheetPanel

Represents a logical part of the grid, such as the column headers, row headers, and scrollable data part.

Constructor

- constructor

Properties

- cellType
- columns
- grid
- height
- hostElement
- rows
- viewRange
- width

Methods

- getCellBoundingRect
- getCellData
- getCellElement
- getSelectedState
- setCellData

Constructor

constructor

```
constructor(g: FlexGrid, cellType: CellType, rows: RowCollection, cols: ColumnCollection, host: HTMLElement): GridPanel
```

Initializes a new instance of the **GridPanel** class.

Parameters

- **g**: **FlexGrid**
  The **FlexGrid** object that owns the panel.
- **cellType**: **CellType**
  The type of cell in the panel.
- **rows**: **RowCollection**
  The rows displayed in the panel.
- **cols**: **ColumnCollection**
  The columns displayed in the panel.
- **host**: **HTMLElement**
  The HTMLElement that hosts the cells in the control.

Returns
GridPanel

Properties

cellType

Gets the type of cell contained in the panel.

Type
CellType

columns

Gets the panel's column collection.

Type
ColumnCollection

grid

Gets the grid that owns the panel.

Type
`FlexGrid`

height

Gets the total height of the content in this panel.

Type
number

hostElement

Gets the host element for the panel.

Type
HTMLElement

rows

Gets the panel's row collection.

Type
`RowCollection`

viewRange

Gets a **`CellRange`** that indicates the range of cells currently visible on the panel.

Type
`CellRange`

width

Gets the total width of the content in the panel.

Type
number

Methods

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a cell's bounds in viewport coordinates.

The returned value is a **Rect** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same as those used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  The index of the row that contains the cell.
- **c**: **number**
  The index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Returns
Rect

getCellData

```
getCellData(r: number, c: any, formatted: boolean): any
```

Gets the value stored in a cell in the panel.

Parameters

- **r**: **number**
  The row index of the cell.
- **c**: **any**
  The index, name, or binding of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Returns
any

getCellElement

```
getCellElement(r: number, c: number): HTMLElement
```

Gets the element that represents a cell within this **GridPanel**.

If the cell is not currently in view, this method returns null.

Parameters

- **r**: **number**
  The index of the row that contains the cell.
- **c**: **number**
  The index of the column that contains the cell.

Returns
HTMLElement

getSelectedState

```
getSelectedState(r: number, c: number, rng: CellRange): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.
- **c**: **number**
  Column index of the cell to inspect.
- **rng**: **CellRange**
  CellRange that contains the cell to inspect.

Returns
SelectedState

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Sets the content of a cell in the panel.

Parameters

- **r**: **number**
  The index of the row that contains the cell.
- **c**: **any**
  The index, name, or binding of the column that contains the cell.
- **value**: **any**
  The value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the grid to show the change.

Returns
boolean

GroupRow Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
Row

Represents a row that serves as a header for a group of rows.

Constructor

- constructor

Properties

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- grid
- hasChildren
- height
- index
- isCollapsed
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- level
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

Methods

- getCellRange
- getGroupHeader
- onPropertyChanged

Constructor

constructor

```
constructor(): GroupRow
```

Initializes a new instance of the **GroupRow** class.

Returns
GroupRow

Properties

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Inherited From
RowCol
Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Inherited From
RowCol
Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Inherited From
RowCol
Type
boolean

collectionView

Gets the **`ICollectionView`** bound to this row or column.

Inherited From
`RowCol`
Type
`ICollectionView`

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Inherited From
`RowCol`
Type
string

dataItem

Gets or sets the item in the data collection that the item is bound to.

Inherited From
`Row`
Type
any

grid

Gets the **`FlexGrid`** that owns the row or column.

Inherited From
`RowCol`
Type
`FlexGrid`

hasChildren

Gets a value that indicates whether the group row has child rows.

Type
boolean

height

Gets or sets the height of the row. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`Row`
Type
number

index

Gets the index of the row or column in the parent collection.

Inherited From
`RowCol`
Type
number

isCollapsed

Gets or sets a value that indicates whether the GroupRow is collapsed (child rows are hidden) or expanded (child rows are visible).

Type
boolean

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Inherited From
`RowCol`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Inherited From
`RowCol`
Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Inherited From
`RowCol`
Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Inherited From
`RowCol`
Type
boolean

level

Gets or sets the hierarchical level of the group associated with the GroupRow.

Type
number

pos

Gets the position of the row or column.

Inherited From
`RowCol`
Type
number

renderHeight

Gets the render height of the row.

The value returned takes into account the row's visibility, default size, and min and max sizes.

Inherited From
`Row`
Type
number

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Inherited From
`RowCol`
Type
number

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`RowCol`
Type
number

visible

Gets or sets a value that indicates whether the row or column is visible.

Inherited From
`RowCol`
Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Inherited From
`RowCol`
Type
number

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Inherited From
`RowCol`
Type
boolean

Methods

getCellRange

```
getCellRange(): CellRange
```

Gets a **`CellRange`** object that contains all of the rows in the group represented by this **`GroupRow`** and all of the columns in the grid.

Returns
`CellRange`

getGroupHeader

```
getGroupHeader(): string
```

Gets the header text for this **GroupRow**.

Returns
string

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Inherited From
RowCol
Returns
void

HitTestInfo Class

File
`wijmo.grid.js`
Module
wijmo.grid

Contains information about the part of a **FlexGrid** control at a given position on the page.

Constructor

- constructor

Properties

- cellType
- col
- edgeBottom
- edgeLeft
- edgeRight
- edgeTop
- grid
- panel
- point
- range
- row

Constructor

constructor

```
constructor(grid: any, pt: any): HitTestInfo
```

Initializes a new instance of the **HitTestInfo** class.

Parameters

- **grid**: **any**
  The **FlexGrid** control, **GridPanel**, or cell element to investigate.

- **pt**: **any**
  The **Point** object in page coordinates to investigate.

Returns
HitTestInfo

Properties

cellType

Gets the type of cell found at the specified position.

Type
`CellType`

col

Gets the column index of the cell at the specified position.

Type
number

edgeBottom

Gets a value that indicates whether the mouse is near the bottom edge of the cell.

Type
boolean

edgeLeft

Gets a value that indicates whether the mouse is near the left edge of the cell.

Type
boolean

edgeRight

Gets a value that indicates whether the mouse is near the right edge of the cell.

Type
boolean

edgeTop

Gets a value that indicates whether the mouse is near the top edge of the cell.

Type
boolean

grid

Gets the **FlexGrid** that this **HitTestInfo** refers to.

Type
FlexGrid

panel

Gets the **GridPanel** that this **HitTestInfo** refers to.

Type
GridPanel

point

Gets the point in control coordinates that this **HitTestInfo** refers to.

Type
Point

range

Gets the cell range at the specified position.

Type
CellRange

row

Gets the row index of the cell at the specified position.

Type
number

MergeManager Class

File
`wijmo.grid.js`
Module
wijmo.grid
Derived Classes
DetailMergeManager

Defines the **FlexGrid**'s cell merging behavior.

An instance of this class is automatically created and assigned to the grid's **mergeManager** property to implement the grid's default merging behavior.

If you want to customize the default merging behavior, create a class that derives from **MergeManager** and override the **getMergedRange** method.

Constructor

- constructor

Methods

- getMergedRange

Constructor

constructor

```
constructor(g: FlexGrid): MergeManager
```

Initializes a new instance of the **MergeManager** class.

Parameters
- **g**: **FlexGrid**
  The **FlexGrid** object that owns this **MergeManager**.

Returns
MergeManager

Methods

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  The index of the row that contains the cell.
- **c**: **number**
  The index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Returns
CellRange

Row Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
RowCol
Derived Classes
GroupRow, **DetailRow**, **HeaderRow**

Represents a row in the grid.

Constructor

- constructor

Properties

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- grid
- height
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

Methods

- onPropertyChanged

Constructor

constructor

```
constructor(dataItem?: any): Row
```

Initializes a new instance of the **Row** class.

Parameters

- **dataItem**: **any** OPTIONAL
  The data item that this row is bound to.

Returns
Row

Properties

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Inherited From
RowCol
Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Inherited From
RowCol
Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Inherited From
RowCol
Type
boolean

collectionView

Gets the **ICollectionView** bound to this row or column.

Inherited From
RowCol
Type
ICollectionView

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Inherited From
RowCol
Type
string

dataItem

Gets or sets the item in the data collection that the item is bound to.

Type
any

grid

Gets the **FlexGrid** that owns the row or column.

Inherited From
RowCol
Type
FlexGrid

height

Gets or sets the height of the row. Setting this property to null or negative values causes the element to use the parent collection's default size.

Type
number

index

Gets the index of the row or column in the parent collection.

Inherited From
`RowCol`
Type
number

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Inherited From
`RowCol`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Inherited From
`RowCol`
Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Inherited From
`RowCol`
Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Inherited From
`RowCol`
Type
boolean

pos

Gets the position of the row or column.

Inherited From
`RowCol`
Type
number

renderHeight

Gets the render height of the row.

The value returned takes into account the row's visibility, default size, and min and max sizes.

Type
number

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Inherited From
`RowCol`
Type
number

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`RowCol`
Type
number

visible

Gets or sets a value that indicates whether the row or column is visible.

Inherited From
`RowCol`
Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Inherited From
`RowCol`
Type
number

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Inherited From
`RowCol`
Type
boolean

Methods

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Inherited From
`RowCol`
Returns
void

RowCol Class

File
`wijmo.grid.js`
Module
wijmo.grid
Derived Classes
Column, **Row**

An abstract class that serves as a base for the **Row** and **Column** classes.

Properties

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- grid
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- pos
- renderSize
- size
- visible
- visibleIndex
- wordWrap

Methods

- onPropertyChanged

Properties

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Type
boolean

collectionView

Gets the **`ICollectionView`** bound to this row or column.

Type
`ICollectionView`

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Type
string

grid

Gets the **`FlexGrid`** that owns the row or column.

Type
`FlexGrid`

index

Gets the index of the row or column in the parent collection.

Type
number

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Type
boolean

pos

Gets the position of the row or column.

Type
number

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Type
number

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Type
number

visible

Gets or sets a value that indicates whether the row or column is visible.

Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Type
number

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Type
boolean

Methods

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Returns
void

RowColCollection Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
ObservableArray
Derived Classes
ColumnCollection, **RowCollection**

Abstract class that serves as a base for row and column collections.

Constructor

- constructor

Properties

- defaultSize
- frozen
- isUpdating
- maxSize
- minSize
- visibleLength

Methods

- beginUpdate
- canMoveElement
- clear
- deferUpdate
- endUpdate
- getItemAt
- getNextCell
- getTotalSize
- implementsInterface
- indexOf
- insert
- isFrozen
- moveElement
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(g: FlexGrid, defaultSize: number): RowColCollection
```

Initializes a new instance of the **RowColCollection** class.

Parameters

- **g**: **FlexGrid**
  The **FlexGrid** that owns the collection.

- **defaultSize**: **number**
  The default size of the elements in the collection.

Returns
RowColCollection

Properties

defaultSize

Gets or sets the default size of elements in the collection.

Type
number

frozen

Gets or sets the number of frozen rows or columns in the collection.

Frozen rows and columns do not scroll, and instead remain at the top or left of the grid, next to the fixed cells. Unlike fixed cells, however, frozen cells may be selected and edited like regular cells.

Type
number

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **beginUpdate** and **endUpdate**).

Inherited From
ObservableArray
Type

maxSize

Gets or sets the maximum size of elements in the collection.

Type
number

minSize

Gets or sets the minimum size of elements in the collection.

Type
number

visibleLength

Gets the number of visible elements in the collection (**`isVisible`**).

Type
number

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Returns
void

canMoveElement

```
canMoveElement(src: number, dst: number): boolean
```

Checks whether an element can be moved from one position to another.

Parameters

- **src**: **number**
  The index of the element to move.

- **dst**: **number**
  The position to which to move the element, or specify -1 to append the element.

Returns
boolean

clear

```
clear(): void
```

Removes all items from the array.

Inherited From
ObservableArray
Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Inherited From
ObservableArray
Returns
void

getItemAt

```
getItemAt(position: number): number
```

Gets the index of the element at a given physical position.

Parameters

- **position**: **number**
  Position of the item in the collection, in pixels.

Returns
number

getNextCell

```
getNextCell(index: number, move: SelMove, pageSize: number): void
```

Finds the next visible cell for a selection change.

Parameters

- **index**: **number**
  Starting index for the search.
- **move**: **SelMove**
  Type of move (size and direction).
- **pageSize**: **number**
  Size of a page (in case the move is a page up/down).

Returns
void

getTotalSize

```
getTotalSize(): number
```

Gets the total size of the elements in the collection.

Returns
number

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
ObservableArray
Returns
boolean

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

Searches for an item in the array.

Parameters

- **searchElement**: **any**
  Element to locate in the array.

- **fromIndex**: **number** OPTIONAL
  The index where the search should start.

Inherited From
ObservableArray
Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Inherited From
ObservableArray
Returns
void

isFrozen

```
isFrozen(index: number): boolean
```

Checks whether a row or column is frozen.

Parameters

- **index**: **number**
  The index of the row or column to check.

Returns
boolean

moveElement

```
moveElement(src: number, dst: number): void
```

Moves an element from one position to another.

Parameters

- **src**: **number**
  Index of the element to move.

- **dst**: **number**
  Position where the element should be moved to (-1 to append).

Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Keeps track of dirty state and invalidate grid on changes.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL

Returns
void

push

```
push(item: any): number
```

Appends an item to the array.

Parameters

- **item**: **any**
  Item to add to the array.

Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
ObservableArray
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Inherited From
ObservableArray
Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
ObservableArray
Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
ObservableArray
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes or adds items to the array.

Parameters

- **index**: **number**
  Position where items are added or removed.

- **count**: **number**
  Number of items to remove from the array.

- **item**: **any** OPTIONAL
  Item to add to the array.

Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

RowCollection Class

File
`wijmo.grid.js`
Module
wijmo.grid
Base Class
RowColCollection

Represents a collection of **Row** objects in a **FlexGrid** control.

Constructor

- constructor

Properties

- defaultSize
- frozen
- isUpdating
- maxGroupLevel
- maxSize
- minSize
- visibleLength

Methods

- beginUpdate
- canMoveElement
- clear
- deferUpdate
- endUpdate
- getItemAt
- getNextCell
- getTotalSize
- implementsInterface
- indexOf
- insert
- isFrozen
- moveElement
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(g: FlexGrid, defaultSize: number): RowColCollection
```

Initializes a new instance of the **RowColCollection** class.

Parameters

- **g**: **FlexGrid**
  The **FlexGrid** that owns the collection.

- **defaultSize**: **number**
  The default size of the elements in the collection.

Inherited From
RowColCollection
Returns
RowColCollection

Properties

defaultSize

Gets or sets the default size of elements in the collection.

Inherited From
RowColCollection
Type
number

frozen

Gets or sets the number of frozen rows or columns in the collection.

Frozen rows and columns do not scroll, and instead remain at the top or left of the grid, next to the fixed cells. Unlike fixed cells, however, frozen cells may be selected and edited like regular cells.

Inherited From
RowColCollection
Type
number

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **`beginUpdate`** and **`endUpdate`**).

Inherited From
`ObservableArray`
Type

maxGroupLevel

Gets the maximum group level in the grid.

Type
number

maxSize

Gets or sets the maximum size of elements in the collection.

Inherited From
`RowColCollection`
Type
number

minSize

Gets or sets the minimum size of elements in the collection.

Inherited From
`RowColCollection`
Type
number

visibleLength

Gets the number of visible elements in the collection (**`isVisible`**).

Inherited From
`RowColCollection`
Type
number

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
RowColCollection
Returns
void

canMoveElement

```
canMoveElement(src: number, dst: number): boolean
```

Checks whether an element can be moved from one position to another.

Parameters

- **src**: **number**
  The index of the element to move.

- **dst**: **number**
  The position to which to move the element, or specify -1 to append the element.

Inherited From
RowColCollection
Returns
boolean

clear

```
clear(): void
```

Removes all items from the array.

Inherited From
ObservableArray
Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Inherited From
ObservableArray
Returns
void

getItemAt

```
getItemAt(position: number): number
```

Gets the index of the element at a given physical position.

Parameters

- **position**: **number**
  Position of the item in the collection, in pixels.

Inherited From
`RowColCollection`
Returns
number

getNextCell

```
getNextCell(index: number, move: SelMove, pageSize: number): void
```

Finds the next visible cell for a selection change.

Parameters

- **index**: **number**
  Starting index for the search.

- **move**: **SelMove**
  Type of move (size and direction).

- **pageSize**: **number**
  Size of a page (in case the move is a page up/down).

Inherited From
`RowColCollection`
Returns
void

getTotalSize

```
getTotalSize(): number
```

Gets the total size of the elements in the collection.

Inherited From
`RowColCollection`
Returns
number

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
`ObservableArray`
Returns
boolean

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

Searches for an item in the array.

Parameters

- **searchElement**: **any**
  Element to locate in the array.

- **fromIndex**: **number** OPTIONAL
  The index where the search should start.

Inherited From
ObservableArray
Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Inherited From
ObservableArray
Returns
void

isFrozen

```
isFrozen(index: number): boolean
```

Checks whether a row or column is frozen.

Parameters

- **index**: **number**
  The index of the row or column to check.

Inherited From
`RowColCollection`
Returns
boolean

moveElement

```
moveElement(src: number, dst: number): void
```

Moves an element from one position to another.

Parameters

- **src**: **number**
  Index of the element to move.

- **dst**: **number**
  Position where the element should be moved to (-1 to append).

Inherited From
`RowColCollection`
Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Keeps track of dirty state and invalidate grid on changes.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL

Inherited From
RowColCollection
Returns
void

push

```
push(item: any): number
```

Appends an item to the array.

Parameters

- **item**: **any**
  Item to add to the array.

Inherited From
RowColCollection
Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
ObservableArray
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Inherited From
ObservableArray
Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
ObservableArray
Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
`ObservableArray`
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes or adds items to the array.

Parameters

- **index**: **number**
  Position where items are added or removed.

- **count**: **number**
  Number of items to remove from the array.

- **item**: **any** OPTIONAL
  Item to add to the array.

Inherited From
`RowColCollection`
Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

AllowDragging Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the row/column dragging behavior.

Members

Name	Value	Description
None	0	The user may not drag rows or columns.
Columns	1	The user may drag columns.
Rows	2	The user may drag rows.
Both	Rows | Columns	The user may drag rows and columns.

AllowMerging Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define which areas of the grid support cell merging.

Members

Name	Value	Description
None	0	No merging.
Cells	1	Merge scrollable cells.
ColumnHeaders	2	Merge column headers.
RowHeaders	4	Merge row headers.
AllHeaders	ColumnHeaders | RowHeaders	Merge column and row headers.
All	Cells | AllHeaders	Merge all areas.

AllowResizing Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the row/column sizing behavior.

Members

Name	Value	Description
None	0	The user may not resize rows or columns.
Columns	1	The user may resize columns by dragging the edge of the column headers.
Rows	2	The user may resize rows by dragging the edge of the row headers.
Both	Rows | Columns	The user may resize rows and columns by dragging the edge of the headers.
ColumnsAllCells	Columns | _AR_ALLCELLS	The user may resize columns by dragging the edge of any cell.
RowsAllCells	Rows | _AR_ALLCELLS	The user may resize rows by dragging the edge of any cell.
BothAllCells	Both | _AR_ALLCELLS	The user may resize rows and columns by dragging the edge of any cell.

AutoSizeMode Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the row/column auto-sizing behavior.

Members

Name	Value	Description
None	0	Autosizing is disabled.
Headers	1	Autosizing accounts for header cells.
Cells	2	Autosizing accounts for data cells.
Both	Headers | Cells	Autosizing accounts for header and data cells.

CellType Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the type of cell in a **GridPanel**.

Members

Name	Value	Description
None	0	Unknown or invalid cell type.
Cell	1	Regular data cell.
ColumnHeader	2	Column header cell.
RowHeader	3	Row header cell.
TopLeft	4	Top-left cell.
ColumnFooter	5	Column footer cell.
BottomLeft	6	Bottom left cell (at the intersection of the row header and column footer cells).

HeadersVisibility Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the visibility of row and column headers.

Members

Name	Value	Description
None	0	No header cells are displayed.
Column	1	Only column header cells are displayed.
Row	2	Only row header cells are displayed.
All	3	Both column and row header cells are displayed.

KeyAction Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the action to perform when special keys such as ENTER and TAB are pressed.

Members

Name	Value	Description
None	0	No special action (let the browser handle the key).
MoveDown	1	Move the selection to the next row.
MoveAcross	2	Move the selection to the next column.
Cycle	3	Move the selection to the next column, then wrap to the next row.
CycleOut	4	Move the selection to the next column, then wrap to the next row, then out of the control.

RowColFlags Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies flags that represent the state of a grid row or column.

Members

Name	Value	Description
Visible	1	The row or column is visible.
AllowResizing	2	The row or column can be resized.
AllowDragging	4	The row or column can be dragged to a new position with the mouse.
AllowMerging	8	The row or column can contain merged cells.
AllowSorting	16	The column can be sorted by clicking its header with the mouse.
AutoGenerated	32	The column was generated automatically.
Collapsed	64	The group row is collapsed.
ParentCollapsed	128	The row has a parent group that is collapsed.
Selected	256	The row or column is selected.
ReadOnly	512	The row or column is read-only (cannot be edited).
HtmlContent	1024	Cells in this row or column contain HTML text.
WordWrap	2048	Cells in this row or column may contain wrapped text.
HasTemplate	4096	Cells in this column have templates.
RowDefault	Visible | AllowResizing	Default settings for new rows.
ColumnDefault	Visible | AllowDragging | AllowResizing | AllowSorting	Default settings for new columns.

SelectedState Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that represent the selected state of a cell.

Members

Name	Value	Description
None	0	The cell is not selected.
Selected	1	The cell is selected but is not the active cell.
Cursor	2	The cell is selected and is the active cell.

SelectionMode Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that define the selection behavior.

Members

Name	Value	Description
None	0	The user cannot select cells using the mouse or keyboard.
Cell	1	The user can select only a single cell at a time.
CellRange	2	The user can select contiguous blocks of cells.
Row	3	The user can select a single row at a time.
RowRange	4	The user can select contiguous rows.
ListBox	5	The user can select non-contiguous rows.

SelMove Enum

File
`wijmo.grid.js`
Module
wijmo.grid

Specifies constants that represent a type of movement for the selection.

Members

Name	Value	Description
None	0	Do not change the selection.
Next	1	Select the next visible cell.
Prev	2	Select the previous visible cell.
NextPage	3	Select the first visible cell in the next page.
PrevPage	4	Select the first visible cell in the previous page.
Home	5	Select the first visible cell.
End	6	Select the last visible cell.
NextCell	7	Select the next visible cell skipping rows if necessary.
PrevCell	8	Select the previous visible cell skipping rows if necessary.

wijmo.grid.filter Module

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter

Extension that provides an Excel-style filtering UI for **FlexGrid** controls.

Classes

- ColumnFilter
- ColumnFilterEditor
- ConditionFilter
- ConditionFilterEditor
- FilterCondition
- FlexGridFilter
- ValueFilter
- ValueFilterEditor

Interfaces

- IColumnFilter

Enums

- FilterType
- Operator

ColumnFilter Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Implements
IColumnFilter

Defines a filter for a column on a **FlexGrid** control.

The **ColumnFilter** contains a **ConditionFilter** and a **ValueFilter**; only one of them may be active at a time.

This class is used by the **FlexGridFilter** class; you rarely use it directly.

Constructor

- constructor

Properties

- column
- conditionFilter
- dataMap
- filterType
- isActive
- valueFilter

Methods

- apply
- clear
- implementsInterface

Constructor

constructor

```
constructor(owner: FlexGridFilter, column: Column): ColumnFilter
```

Initializes a new instance of the **ColumnFilter** class.

Parameters

- **owner**: **FlexGridFilter**
  The **FlexGridFilter** that owns this column filter.

- **column**: **Column**
  The **Column** to filter.

Returns
ColumnFilter

Properties

column

Gets the **Column** being filtered.

Type
Column

conditionFilter

Gets the **ConditionFilter** in this **ColumnFilter**.

Type
ConditionFilter

dataMap

Gets or sets the **DataMap** used to convert raw values into display values shown when editing this filter.

The example below assigns a **DataMap** to Boolean column filters so the filter editor displays 'Yes' and 'No' instead of 'true' and 'false':

```
var filter = new wijmo.grid.filter.FlexGridFilter(grid),
    map = new wijmo.grid.DataMap([
            { value: true, caption: 'Yes' },
            { value: false, caption: 'No' },
        ], 'value', 'caption');
for (var c = 0; c < grid.columns.length; c++) {
    if (grid.columns[c].dataType == wijmo.DataType.Boolean) {
        filter.getColumnFilter(c).dataMap = map;
    }
}
```

Type
DataMap

filterType

Gets or sets the types of filtering provided by this filter.

Setting this property to null causes the filter to use the value defined by the owner filter's **defaultFilterType** property.

Type
FilterType

isActive

Gets a value that indicates whether the filter is active.

Type
boolean

valueFilter

Gets the **ValueFilter** in this **ColumnFilter**.

Type
ValueFilter

Methods

apply

```
apply(value): boolean
```

Gets a value that indicates whether a value passes the filter.

Parameters

- **value**:
  The value to test.

Returns
boolean

clear

```
clear(): void
```

Clears the filter.

Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Returns
boolean

ColumnFilterEditor Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Base Class
Control

The editor used to inspect and modify column filters.

This class is used by the **FlexGridFilter** class; you rarely use it directly.

Constructor

- constructor

Properties

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onButtonClicked
- onFilterChanged
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

Events

- buttonClicked
- filterChanged
- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, filter: ColumnFilter, sortButtons?: boolean): ColumnFilterEditor
```

Initializes a new instance of the **`ColumnFilterEditor`** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **filter**: **ColumnFilter**
  The **`ColumnFilter`** to edit.
- **sortButtons**: **boolean** OPTIONAL
  Whether to show sort buttons in the editor.

Returns
ColumnFilterEditor

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **`ColumnFilterEditor`** controls.

Type
any

filter

Gets a reference to the **`ColumnFilter`** being edited.

Type
`ColumnFilter`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onButtonClicked

```
onButtonClicked(e?: EventArgs): void
```

Raises the **buttonClicked** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onFilterChanged

```
onFilterChanged(e?: EventArgs): void
```

Raises the **filterChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`Control`
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

updateEditor

```
updateEditor(): void
```

Updates editor with current filter settings.

Returns
void

updateFilter

```
updateFilter(): void
```

Updates filter with current editor settings.

Returns
void

Events

buttonClicked

Occurs when one of the editor buttons is clicked.

Arguments
EventArgs

filterChanged

Occurs after the filter is modified.

Arguments
EventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

ConditionFilter Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Implements
IColumnFilter

Defines a condition filter for a column on a **FlexGrid** control.

Condition filters contain two conditions that may be combined using an 'and' or an 'or' operator.

This class is used by the **FlexGridFilter** class; you will rarely use it directly.

Constructor

- constructor

Properties

- and
- column
- condition1
- condition2
- dataMap
- isActive

Methods

- apply
- clear
- implementsInterface

Constructor

constructor

```
constructor(column: Column): ConditionFilter
```

Initializes a new instance of the **ConditionFilter** class.

Parameters
- **column**: **Column**
  The column to filter.

Returns
ConditionFilter

Properties

and

Gets a value that indicates whether to combine the two conditions with an AND or an OR operator.

Type
boolean

column

Gets the **`Column`** to filter.

Type
`Column`

condition1

Gets the first condition in the filter.

Type
`FilterCondition`

condition2

Gets the second condition in the filter.

Type
`FilterCondition`

dataMap

Gets or sets the **`DataMap`** used to convert raw values into display values shown when editing this filter.

Type
`DataMap`

isActive

Gets a value that indicates whether the filter is active.

The filter is active if at least one of the two conditions has its operator and value set to a valid combination.

Type
boolean

Methods

apply

```
apply(value): boolean
```

Returns a value indicating whether a value passes this filter.

Parameters

- **value**:
  The value to test.

Returns
boolean

clear

```
clear(): void
```

Clears the filter.

Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Returns
boolean

ConditionFilterEditor Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Base Class
Control

The editor used to inspect and modify **ConditionFilter** objects.

This class is used by the **FlexGridFilter** class; you rarely use it directly.

Constructor

- constructor

Properties

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, filter: ConditionFilter): ConditionFilterEditor
```

Initializes a new instance of the **ConditionFilterEditor** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **filter**: **ConditionFilter**
  The **ConditionFilter** to edit.

Returns
ConditionFilterEditor

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **ConditionFilterEditor** controls.

Type
any

filter

Gets a reference to the **ConditionFilter** being edited.

Type
ConditionFilter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

clearEditor

```
clearEditor(): void
```

Clears the editor without applying changes to the filter.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

updateEditor

```
updateEditor(): void
```

Updates editor with current filter settings.

Returns
void

updateFilter

```
updateFilter(): void
```

Updates filter to reflect the current editor values.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

FilterCondition Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter

Defines a filter condition.

This class is used by the **FlexGridFilter** class; you will rarely have to use it directly.

Properties

- isActive
- operator
- value

Methods

- apply
- clear

Properties

isActive

Gets a value that indicates whether the condition is active.

Type
boolean

operator

Gets or sets the operator used by this **`FilterCondition`**.

Type
`Operator`

value

Gets or sets the value used by this **`FilterCondition`**.

Type
any

Methods

apply

```
apply(value): boolean
```

Returns a value that determines whether the given value passes this **`FilterCondition`**.

Parameters

- **value**:
  The value to test.

Returns
boolean

clear

```
clear(): void
```

Clears the condition.

Returns
void

FlexGridFilter Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Derived Classes
WjFlexGridFilter

Implements an Excel-style filter for **FlexGrid** controls.

To enable filtering on a **FlexGrid** control, create an instance of the **FlexGridFilter** and pass the grid as a parameter to the constructor. For example:

```
// create FlexGrid
var flex = new wijmo.grid.FlexGrid('#gridElement');

// enable filtering on the FlexGrid
var filter = new wijmo.grid.filter.FlexGridFilter(flex);
```

Once this is done, a filter icon is added to the grid's column headers. Clicking the icon shows an editor where the user can edit the filter conditions for that column.

The **FlexGridFilter** class depends on the **wijmo.grid** and **wijmo.input** modules.

Constructor

- constructor

Properties

- defaultFilterType
- filterColumns
- filterDefinition
- grid
- showFilterIcons
- showSortButtons

Methods

- apply
- clear
- closeEditor
- editColumnFilter
- getColumnFilter
- onFilterApplied
- onFilterChanged
- onFilterChanging

Events

- filterApplied
- filterChanged
- filterChanging

Constructor

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridFilter
```

Initializes a new instance of the **FlexGridFilter** class.

Parameters

- **grid**: **FlexGrid**
  The **FlexGrid** to filter.

- **options**: **any** OPTIONAL
  Initialization options for the **FlexGridFilter**.

Returns
FlexGridFilter

Properties

defaultFilterType

Gets or sets the default filter type to use.

This value can be overridden in filters for specific columns. For example, the code below creates a filter that filters by conditions on all columns except the "ByValue" column:

```
var f = new wijmo.grid.filter.FlexGridFilter(flex);
f.defaultFilterType = wijmo.grid.filter.FilterType.Condition;
var col = flex.columns.getColumn('ByValue'),
    cf = f.getColumnFilter(col);
cf.filterType = wijmo.grid.filter.FilterType.Value;
```

Type
FilterType

filterColumns

Gets or sets an array containing the names or bindings of the columns that have filters.

Setting this property to null or to an empty array adds filters to all columns.

Type
string[]

filterDefinition

Gets or sets the current filter definition as a JSON string.

Type
string

grid

Gets a reference to the **FlexGrid** that owns this filter.

Type
FlexGrid

showFilterIcons

Gets or sets a value indicating whether the **FlexGridFilter** adds filter editing buttons to the grid's column headers.

If you set this property to false, then you are responsible for providing a way for users to edit, clear, and apply the filters.

Type
boolean

showSortButtons

Gets or sets a value indicating whether the filter editor should include sort buttons.

By default, the editor shows sort buttons like Excel does. But since users can sort columns by clicking their headers, sort buttons in the filter editor may not be desirable in some circumstances.

Type
boolean

Methods

apply

```
apply(): void
```

Applies the current column filters to the grid.

Returns
void

clear

```
clear(): void
```

Clears all column filters.

Returns
void

closeEditor

```
closeEditor(): void
```

Closes the filter editor.

Returns
void

editColumnFilter

```
editColumnFilter(col: any, ht?: HitTestInfo): void
```

Shows the filter editor for the given grid column.

Parameters

- **col**: **any**
  The **`Column`** that contains the filter to edit.

- **ht**: **HitTestInfo** OPTIONAL
  A **`HitTestInfo`** object containing the range of the cell that triggered the filter display.

Returns
void

getColumnFilter

```
getColumnFilter(col: any, create?: boolean): ColumnFilter
```

Gets the filter for the given column.

Parameters

- **col**: **any**
  The **`Column`** that the filter applies to (or column name or index).

- **create**: **boolean** OPTIONAL
  Whether to create the filter if it does not exist.

Returns
`ColumnFilter`

onFilterApplied

```
onFilterApplied(e?: EventArgs): void
```

Raises the **filterApplied** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onFilterChanged

```
onFilterChanged(e: CellRangeEventArgs): void
```

Raises the **filterChanged** event.

Parameters

- **e**: **CellRangeEventArgs**

Returns
void

onFilterChanging

```
onFilterChanging(e: CellRangeEventArgs): void
```

Raises the **filterChanging** event.

Parameters

- **e**: **CellRangeEventArgs**

Returns
void

Events

filterApplied

Occurs after the filter is applied.

Arguments
`EventArgs`

filterChanged

Occurs after a column filter has been edited by the user.

Use the event parameters to determine the column that owns the filter and whether changes were applied or canceled.

Arguments
`CellRangeEventArgs`

filterChanging

Occurs when a column filter is about to be edited by the user.

Use this event to customize the column filter if you want to override the default settings for the filter.

For example, the code below sets the operator used by the filter conditions to 'contains' if they are null:

```
filter.filterChanging.addHandler(function (s, e) {
  var cf = filter.getColumnFilter(e.col);
  if (!cf.valueFilter.isActive && cf.conditionFilter.condition1.operator == null) {
    cf.filterType = wijmo.grid.filter.FilterType.Condition;
    cf.conditionFilter.condition1.operator = wijmo.grid.filter.Operator.CT;
  }
});
```

Arguments
`CellRangeEventArgs`

ValueFilter Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Implements
IColumnFilter

Defines a value filter for a column on a **FlexGrid** control.

Value filters contain an explicit list of values that should be displayed by the grid.

Constructor

- constructor

Properties

- column
- dataMap
- filterText
- isActive
- maxValues
- showValues
- sortValues
- uniqueValues

Methods

- apply
- clear
- implementsInterface

Constructor

constructor

```
constructor(column: Column): ValueFilter
```

Initializes a new instance of the **ValueFilter** class.

Parameters

- **column**: **Column**
  The column to filter.

Returns
ValueFilter

Properties

column

Gets the **`Column`** to filter.

Type
`Column`

dataMap

Gets or sets the **`DataMap`** used to convert raw values into display values shown when editing this filter.

Type
`DataMap`

filterText

Gets or sets a string used to filter the list of display values.

Type
string

isActive

Gets a value that indicates whether the filter is active.

The filter is active if there is at least one value is selected.

Type
boolean

maxValues

Gets or sets the maximum number of elements on the list of display values.

Adding too many items to the list makes searching difficult and hurts performance. This property limits the number of items displayed at any time, but users can still use the search box to filter the items they are interested in.

This property is set to 250 by default.

This code changes the value to 1,000,000, effectively listing all unique values for the field:

```
// change the maxItems property for the 'id' column:
var f = new wijmo.grid.filter.FlexGridFilter(s);
f.getColumnFilter('id').valueFilter.maxValues = 1000000;
```

Type
number

showValues

Gets or sets an object with all the formatted values that should be shown on the value list.

Type
any

sortValues

Gets or sets a value that determines whether the values should be sorted when displayed in the editor.

This property is especially useful when you are using the **`uniqueValues`** to provide a custom list of values property and you would like to preserve the order of the values.

Type
boolean

uniqueValues

Gets or sets an array containing the unique values to be displayed on the list.

If this property is set to null, the list will be filled based on the grid data.

Explicitly assigning the list of unique values is more efficient than building the list from the data, and is required for value filters to work properly when the data is filtered on the server (because in this case some values might not be present on the client so the list will be incomplete).

By default, the filter editor will sort the unique values when displaying them to the user. If you want to prevent that and show the values in the order you provided, set the **`sortValues`** property to false.

For example, the code below provides a list of countries to be used in the **`ValueFilter`** for the column bound to the 'country' field:

```
// create filter for a FlexGrid
var filter = new wijmo.grid.filter.FlexGridFilter(grid);

// assign list of unique values to country filter
var cf = filter.getColumnFilter('country');
cf.valueFilter.uniqueValues = countries;
```

Type
any[]

Methods

apply

```
apply(value): boolean
```

Gets a value that indicates whether a value passes the filter.

Parameters

- **value**:
  The value to test.

Returns
boolean

clear

```
clear(): void
```

Clears the filter.

Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Returns
boolean

ValueFilterEditor Class

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter
Base Class
Control

The editor used to inspect and modify **ValueFilter** objects.

This class is used by the **FlexGridFilter** class; you rarely use it directly.

Constructor

- constructor

Properties

- controlTemplate
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, filter: ValueFilter): ValueFilterEditor
```

Initializes a new instance of the **ValueFilterEditor** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **filter**: **ValueFilter**
  The **ValueFilter** to edit.

Returns
ValueFilterEditor

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **ColumnFilterEditor** controls.

Type
any

filter

Gets a reference to the **ValueFilter** being edited.

Type
ValueFilter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

clearEditor

```
clearEditor(): void
```

Clears the editor without applying changes to the filter.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

updateEditor

```
updateEditor(): void
```

Updates editor with current filter settings.

Returns
void

updateFilter

```
updateFilter(): void
```

Updates filter to reflect the current editor values.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

IColumnFilter Interface

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter

Defines a filter for a column on a **FlexGrid** control.

This class is used by the **FlexGridFilter** class; you rarely use it directly.

FilterType Enum

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter

Specifies types of column filter.

Members

Name	Value	Description
None	0	No filter.
Condition	1	A filter based on two conditions.
Value	2	A filter based on a set of values.
Both	3	A filter that combines condition and value filters.

Operator Enum

File
`wijmo.grid.filter.js`
Module
wijmo.grid.filter

Specifies filter condition operators.

Members

Name	Value	Description
EQ	0	Equals.
NE	1	Does not equal.
GT	2	Greater than.
GE	3	Greater than or equal to.
LT	4	Less than.
LE	5	Less than or equal to.
BW	6	Begins with.
EW	7	Ends with.
CT	8	Contains.
NC	9	Does not contain.

wijmo.grid.grouppanel Module

File
`wijmo.grid.grouppanel.js`
Module
wijmo.grid.grouppanel

Extension that provides a drag and drop UI for editing groups in bound **FlexGrid** controls.

Classes

- GroupPanel

GroupPanel Class

File
`wijmo.grid.grouppanel.js`
Module
wijmo.grid.grouppanel
Base Class
Control
Derived Classes
WjGroupPanel

The **GroupPanel** control provides a drag and drop UI for editing groups in a bound **FlexGrid** control.

It allows users to drag columns from the **FlexGrid** into the panel and to move groups within the panel. Users may click the group markers in the panel to sort based on the group column or to remove groups.

In order to use a **GroupPanel**, add it to a page that contains a **FlexGrid** control and set the panel's **grid** property to the **FlexGrid** control. For example:

```
// create a FlexGrid
var flex = new wijmo.grid.FlexGrid('#flex-grid');
flex.itemsSource = getData();

// add a GroupPanel to edit data groups
var groupPanel = new wijmo.grid.grouppanel.GroupPanel('#group-panel');
groupPanel.placeholder = "Drag columns here to create groups.";
groupPanel.grid = flex;
```

Constructor

- constructor

Properties

- controlTemplate
- grid
- hideGroupedColumns
- hostElement
- isDisabled
- isTouching
- isUpdating
- maxGroups
- placeholder
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, options?): GroupPanel
```

Initializes a new instance of the **GroupPanel** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
GroupPanel

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **GroupPanel** controls.

Type
any

grid

Gets or sets the **FlexGrid** that is connected to this **GroupPanel**.

Once a grid is connected to the panel, the panel displays the groups defined in the grid's data source. Users can drag grid columns into the panel to create new groups, drag groups within the panel to re-arrange the groups, or delete items in the panel to remove the groups.

Type
FlexGrid

hideGroupedColumns

Gets or sets a value indicating whether the panel hides grouped columns in the owner grid.

The **`FlexGrid`** displays grouping information in row headers, so it is usually a good idea to hide grouped columns since they display redundant information.

Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

maxGroups

Gets or sets the maximum number of groups allowed.

Type
number

placeholder

Gets or sets a string to display in the control when it contains no groups.

Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(): void
```

Updates the panel to show the current groups.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

wijmo.grid.detail Module

File
`wijmo.grid.detail.js`
Module
wijmo.grid.detail

Extension that provides detail rows for **FlexGrid** controls.

Classes

- DetailMergeManager
- DetailRow
- FlexGridDetailProvider

Enums

- DetailVisibilityMode

DetailMergeManager Class

File
`wijmo.grid.detail.js`
Module
wijmo.grid.detail
Base Class
MergeManager

Merge manager class used by the **FlexGridDetailProvider** class.

The **DetailMergeManager** merges detail cells (cells in a **DetailRow**) into a single detail cell that spans all grid columns.

Constructor

- constructor

Methods

- getMergedRange

Constructor

constructor

```
constructor(grid: FlexGrid): DetailMergeManager
```

Initializes a new instance of the **DetailMergeManager** class.

Parameters

- **grid**: **FlexGrid**
  The **FlexGrid** object that owns this **DetailMergeManager**.

Returns
DetailMergeManager

Methods

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  The index of the row that contains the cell.
- **c**: **number**
  The index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Returns
CellRange

DetailRow Class

File
`wijmo.grid.detail.js`
Module
wijmo.grid.detail
Base Class
Row

Row that contains a single detail cell spanning all grid columns.

Constructor

- constructor

Properties

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- detail
- grid
- height
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

Methods

- onPropertyChanged

Constructor

constructor

```
constructor(parentRow: Row): DetailRow
```

Initializes a new instance of the **DetailRow** class.

Parameters

- **parentRow**: **Row**
  Row that this **DetailRow** provides details for.

Returns
DetailRow

Properties

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Inherited From
RowCol
Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Inherited From
RowCol
Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Inherited From
RowCol
Type
boolean

collectionView

Gets the **`ICollectionView`** bound to this row or column.

Inherited From
`RowCol`
Type
`ICollectionView`

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Inherited From
`RowCol`
Type
string

dataItem

Gets or sets the item in the data collection that the item is bound to.

Inherited From
`Row`
Type
any

detail

Gets or sets the HTML element that represents the detail cell in this **`DetailRow`**.

Type
HTMLElement

grid

Gets the **FlexGrid** that owns the row or column.

Inherited From
RowCol
Type
FlexGrid

height

Gets or sets the height of the row. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
Row
Type
number

index

Gets the index of the row or column in the parent collection.

Inherited From
RowCol
Type
number

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Inherited From
RowCol
Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Inherited From
`RowCol`
Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Inherited From
`RowCol`
Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Inherited From
`RowCol`
Type
boolean

pos

Gets the position of the row or column.

Inherited From
`RowCol`
Type
number

renderHeight

Gets the render height of the row.

The value returned takes into account the row's visibility, default size, and min and max sizes.

Inherited From
`Row`
Type
number

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Inherited From
`RowCol`
Type
number

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`RowCol`
Type
number

visible

Gets or sets a value that indicates whether the row or column is visible.

Inherited From
`RowCol`
Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Inherited From
`RowCol`
Type
number

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Inherited From
`RowCol`
Type
boolean

Methods

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Inherited From
`RowCol`
Returns
void

FlexGridDetailProvider Class

File
`wijmo.grid.detail.js`
Module
wijmo.grid.detail
Derived Classes
WjFlexGridDetail

Implements detail rows for **FlexGrid** controls.

To add detail rows to a **FlexGrid** control, create an instance of a **FlexGridDetailProvider** and set the **createDetailCell** property to a function that creates elements to be displayed in the detail cells.

For example:

```
// create FlexGrid to show categories
var gridCat = new wijmo.grid.FlexGrid('#gridCat');
gridCat.itemsSource = getCategories();

// add detail rows showing products in each category
var detailProvider = new wijmo.grid.detail.FlexGridDetailProvider(gridCat);
detailProvider.createDetailCell = function (row) {
  var cell = document.createElement('div');
  var gridProducts = new wijmo.grid.FlexGrid(cell);
  gridProducts.itemsSource = getProducts(row.dataItem.CategoryID);
  return cell;
}
```

The **FlexGridDetailProvider** provides a **detailVisibilityMode** property that determines when the detail rows should be displayed. The default value for this property is **ExpandSingle**, which adds collapse/expand icons to the row headers.

Constructor

- constructor

Properties

- createDetailCell
- detailVisibilityMode
- disposeDetailCell
- grid
- isAnimated
- maxHeight
- rowHasDetail

Methods

- getDetailRow
- hideDetail
- isDetailAvailable
- isDetailVisible
- showDetail

Constructor

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridDetailProvider
```

Initializes a new instance of the **FlexGridDetailProvider** class.

Parameters

- **grid**: **FlexGrid**
  FlexGrid that will receive detail rows.

- **options**: **any** OPTIONAL
  Initialization options for the new **FlexGridDetailProvider**.

Returns
FlexGridDetailProvider

Properties

createDetailCell

Gets or sets the callback function that creates detail cells.

The callback function takes a **Row** as a parameter and returns an HTML element representing the row details. For example:

```
// create detail cells for a given row
dp.createDetailCell = function (row) {
  var cell = document.createElement('div');
  var detailGrid = new wijmo.grid.FlexGrid(cell, {
    itemsSource: getProducts(row.dataItem.CategoryID),
    headersVisibility: wijmo.grid.HeadersVisibility.Column
  });
  return cell;
};
```

Type
Function

detailVisibilityMode

Gets or sets a value that determines when row details are displayed.

Type
DetailVisibilityMode

disposeDetailCell

Gets or sets the callback function that disposes of detail cells.

The callback function takes a **Row** as a parameter and disposes of any resources associated with the detail cell.

This function is optional. Use it in cases where the **createDetailCell** function allocates resources that are not automatically garbage-collected.

Type
Function

grid

Gets the **FlexGrid** that owns this **FlexGridDetailProvider**.

Type
FlexGrid

isAnimated

Gets or sets a value that indicates whether to use animation when showing row details.

Type
boolean

maxHeight

Gets or sets the maximum height of the detail rows, in pixels.

Type
number

rowHasDetail

Gets or sets the callback function that determines whether a row has details.

The callback function takes a **Row** as a parameter and returns a boolean value that indicates whether the row has details. For example:

```
// remove details from items with odd CategoryID
dp.rowHasDetail = function (row) {
  return row.dataItem.CategoryID % 2 == 0;
};
```

Setting this property to null indicates all rows have details.

Type
Function

Methods

getDetailRow

```
getDetailRow(row: any): DetailRow
```

Gets the detail row associated with a given grid row.

Parameters

- **row**: **any**
  Row or index of the row to investigate.

Returns
`DetailRow`

hideDetail

```
hideDetail(row?: any): void
```

Hides the detail row for a given row.

Parameters

- **row**: **any** OPTIONAL
  Row or index of the row that will have its details hidden. This parameter is optional. If not provided, all detail rows are hidden.

Returns
void

isDetailAvailable

```
isDetailAvailable(row: any): boolean
```

Gets a value that determines if a row has details to show.

Parameters

- **row**: **any**
  Row or index of the row to investigate.

Returns
boolean

isDetailVisible

```
isDetailVisible(row: any): boolean
```

Gets a value that determines if a row's details are visible.

Parameters

- **row**: **any**
  Row or index of the row to investigate.

Returns
boolean

showDetail

```
showDetail(row: any, hideOthers?: boolean): void
```

Shows the detail row for a given row.

Parameters

- **row**: **any**
  Row or index of the row that will have its details shown.

- **hideOthers**: **boolean** OPTIONAL
  Whether to hide details for all other rows.

Returns
void

DetailVisibilityMode Enum

File
`wijmo.grid.detail.js`
Module
wijmo.grid.detail

Specifies when and how the row details are displayed.

Members

Name	Value	Description
Code	0	Details are shown or hidden in code, using the **`showDetail`** and **`hideDetail`** methods.
Selection	1	Details are shown for the row that is currently selected.
ExpandSingle	2	Details are shown or hidden using buttons added to the row headers. Only one row may be expanded at a time.
ExpandMulti	3	Details are shown or hidden using buttons added to the row headers. Multiple rows may be expanded at a time.

wijmo.grid.xlsx Module

File
`wijmo.grid.xlsx.js`
Module
wijmo.grid.xlsx

Extension that defines the **FlexGridXlsxConverter** class that provides client-side Excel xlsx file save/load capabilities for the **FlexGrid** control.

Classes

- FlexGridXlsxConverter
- XlsxFormatItemEventArgs

Interfaces

- IExtendedSheetInfo
- IFlexGridXlsxOptions

FlexGridXlsxConverter Class

File
`wijmo.grid.xlsx.js`
Module
wijmo.grid.xlsx

This class provides static **load** and **save** methods for loading and saving **FlexGrid** controls from and to Excel xlsx files.

Methods

- load
- loadAsync
- save
- saveAsync

Methods

STATIC load

```
load(grid: FlexGrid, workbook: any, options?: IFlexGridXlsxOptions): void
```

Loads a **Workbook** instance or a Blob object containing xlsx file content to the **FlexGrid** instance. This method works with JSZip 2.5.

For example:

```
// This sample opens an xlsx file chosen through Open File
// dialog and fills FlexGrid with the content of the first

// sheet.


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>
<div id="flexHost"></>


// JavaScript
var flexGrid = new wijmo.grid.FlexGrid("#flexHost"),
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
            wijmo.grid.xlsx.FlexGridXlsxConverter.load(flexGrid, reader.result,
                { includeColumnHeaders: true });
        };
        reader.readAsArrayBuffer(file);
    }
}
```

Parameters

- **grid**: **FlexGrid**
  FlexGrid that loads the **Workbook** object.

- **workbook**: **any**
  A **Workbook**, Blob, base-64 string, or ArrayBuffer containing the xlsx file content.

- **options**: **IFlexGridXlsxOptions** OPTIONAL

IFlexGridXlsxOptions object specifying the load options.

Returns
void

STATIC loadAsync

```
loadAsync(grid: FlexGrid, workbook: any, options?: IFlexGridXlsxOptions, onLoaded?: Workbook), onError?: (reason?: any)): void
```

Asynchronously loads a **Workbook** or a Blob representing an xlsx file into a **FlexGrid**.

This method requires JSZip 3.0.

Parameters

- **grid**: **FlexGrid**
  FlexGrid that loads the **Workbook** object.
- **workbook**: **any**
  Workbook, Blob, base-64 string, or ArrayBuffer representing the xlsx file content.
- **options**: **IFlexGridXlsxOptions** OPTIONAL
  IFlexGridXlsxOptions object specifying the load options.
- **onLoaded**: **(workbook: wijmo.xlsx.Workbook)** OPTIONAL
  Callback invoked when the method finishes executing. The callback provides access to the workbook that was loaded (passed as a parameter to the callback).
- **onError**: **(reason?: any)** OPTIONAL
  Callback invoked when there are errors saving the file. The error is passed as a parameter to the callback.

  For example:

```
wijmo.grid.xlsx.FlexGridXlsxConverter.loadAsync(grid, blob, null, function (workbook) {

     // user can access the loaded workbook instance in this callback.
     var app = worksheet.application ;
     ...
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of save failure is ' + reason);
});
```

Returns
void

STATIC save

```
save(grid: FlexGrid, options?: IFlexGridXlsxOptions, fileName?: string): Workbook
```

Save the **FlexGrid** instance to the **Workbook** instance. This method works with JSZip 2.5.

For example:

```
// This sample exports FlexGrid content to an xlsx file.
// click.


// HTML
<button
    onclick="saveXlsx('FlexGrid.xlsx')">
    Save
</button>


// JavaScript
function saveXlsx(fileName) {

    // Save the flexGrid to xlsx file.
    wijmo.grid.xlsx.FlexGridXlsxConverter.save(flexGrid,
            { includeColumnHeaders: true }, fileName);
}
```

Parameters

- **grid**: **FlexGrid**
  FlexGrid that will be saved.

- **options**: **IFlexGridXlsxOptions** OPTIONAL
  IFlexGridXlsxOptions object specifying the save options.

- **fileName**: **string** OPTIONAL
  Name of the file that will be generated.

Returns
Workbook

STATIC **saveAsync**

```
saveAsync(grid: FlexGrid, options?: IFlexGridXlsxOptions, fileName?: string, onSaved?: (base64: string), onError?: (reason?: any)): Workbook
```

Asynchronously saves the content of a **FlexGrid** to a file.

This method requires JSZip 3.0.

Parameters

- **grid**: **FlexGrid**
  FlexGrid that will be saved.

- **options**: **IFlexGridXlsxOptions** OPTIONAL
  IFlexGridXlsxOptions object specifying the save options.

- **fileName**: **string** OPTIONAL
  Name of the file that will be generated.

- **onSaved**: **(base64: string)** OPTIONAL
  Callback invoked when the method finishes executing. The callback provides access to the content of the saved workbook (encoded as a base-64 string and passed as a parameter to the callback).

- **onError**: **(reason?: any)** OPTIONAL
  Callback invoked when there are errors saving the file. The error is passed as a parameter to the callback.

  For example:

```
wijmo.grid.xlsx.FlexGridXlsxConverter.saveAsync(flexGrid,
    { includeColumnHeaders: true }, // options
    'FlexGrid.xlsx', // filename
    function (base64) { // onSaved
        // User can access the base64 string in this callback.
        document.getElementById('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64,' + base64;
    },
    function (reason) { // onError
        // User can catch the failure reason in this callback.
        console.log('The reason of save failure is ' + reason);
    }
);
```

Returns
Workbook

XlsxFormatItemEventArgs Class

File
`wijmo.grid.xlsx.js`
Module
wijmo.grid.xlsx
Base Class
CellRangeEventArgs

Represents arguments of the IFlexGridXlsxOptions.formatItem callback.

Constructor

- constructor

Properties

- cancel
- cell
- col
- data
- empty
- panel
- range
- row
- xlsxCell

Methods

- getFormattedCell

Constructor

constructor

```
constructor(p: GridPanel, rng: CellRange, data?: any): CellRangeEventArgs
```

Initializes a new instance of the **CellRangeEventArgs** class.

Parameters

- **p**: **GridPanel**
  GridPanel that contains the range.
- **rng**: **CellRange**
  Range of cells affected by the event.
- **data**: **any** OPTIONAL
  Data related to the event.

Inherited From
CellRangeEventArgs
Returns
CellRangeEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
CancelEventArgs
Type
boolean

cell

If IFlexGridXlsxOptions.includeCellStyles is set to true then contains a reference to the element that represents the formatted grid cell; otherwise, a null value.

Type
HTMLElement

col

Gets the column affected by this event.

Inherited From
`CellRangeEventArgs`
Type
number

data

Gets or sets the data associated with the event.

Inherited From
`CellRangeEventArgs`
Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

panel

Gets the **GridPanel** affected by this event.

Inherited From
`CellRangeEventArgs`
Type
`GridPanel`

range

Gets the **CellRange** affected by this event.

Inherited From
CellRangeEventArgs
Type
CellRange

row

Gets the row affected by this event.

Inherited From
CellRangeEventArgs
Type
number

xlsxCell

Contains an exporting cell representation. Initially it contains a default cell representation created by FlexGrid export, and can be modified by the event handler to customize its final content. For example, the xlsxCell.value property can be updated to modify a cell content, xlsxCell.style to modify cell's style, and so on.

Type
IWorkbookCell

Methods

getFormattedCell

```
getFormattedCell(): HTMLElement
```

Returns a cell with a custom formatting applied (formatItem event, cell templates). This method is useful when export of custom formatting is disabled (IFlexGridXlsxOptions.includeCellStyles=false), but you need to export a custom content and/or style for a certain cells.

Returns
HTMLElement

IExtendedSheetInfo Interface

File
`wijmo.grid.xlsx.js`
Module
wijmo.grid.xlsx

Defines additional worksheet properties that can be accesses via the dynamic **wj_sheetInfo** property of the **FlexGrid** instance.

Properties

- fonts
- mergedRanges
- name
- styledCells
- tableNames
- visible

Properties

fonts

Contains an array of font names used in the sheet.

Type
string[]

mergedRanges

Merged ranges in the sheet

Type
any

name

The sheet name.

Type
string

styledCells

Styled cells in the sheet

Type
any

tableNames

The name of tables refered in this worksheet.

Type
string[]

visible

Sheet visibility.

Type
boolean

IFlexGridXlsxOptions Interface

File
`wijmo.grid.xlsx.js`
Module
wijmo.grid.xlsx

FlexGrid Xlsx conversion options

Properties

Properties

activeWorksheet

Index or name of the active sheet in the xlsx file.

Type
any

formatItem

An optional callback which is called for every exported cell and allows to perform transformations of exported cell value and style. The callback is called irrespectively of the 'includeCellStyles' property value.

Type

includeCellStyles

Indicates whether cells styling should be included in the generated xlsx file.

Type
boolean

includeColumnHeaders

Indicates whether to include column headers as first rows in the generated xlsx file.

Type
boolean

includeColumns

A callback to indicate which columns of FlexGrid need be included or omitted during exporting.

For example:

```
// This sample excludes the 'country' column from export.


// JavaScript
wijmo.grid.xlsx.FlexGridXlsxConverter.save(grid, {
  includeColumns: function(column) {
     return column.binding !== 'country';
  }
}
```

Type

includeRowHeaders

Indicates whether to include column headers as first rows in the generated xlsx file.

Type
boolean

sheetIndex

The index of the sheet in the workbook. It indicates to import which sheet.

Type
number

sheetName

The name of the sheet. It indicates to import which sheet for importing. If the sheetIndex and sheetName are both setting, the priority of sheetName is higher than sheetIndex. It sets the name of worksheet for exporting.

Type
string

sheetVisible

The visible of the sheet.

Type
boolean

wijmo.grid.multirow Module

File
`wijmo.grid.multirow.js`
Module
wijmo.grid.multirow

Defines the **MultiRow** control and its associated classes.

Classes

- MultiRow

MultiRow Class

File
`wijmo.grid.multirow.js`
Module
wijmo.grid.multirow
Base Class
FlexGrid
Derived Classes
WjMultiRow

Extends the **FlexGrid** control to provide multiple rows per item.

Use the **layoutDefinition** property to define the layout of the rows used to display each data item.

A few **FlexGrid** properties are disabled in the **MultiRow** control because they would interfere with the custom multi-row layouts. The list of disabled properties includes **allowMerging** and **childItemsPath**.

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsedHeaders
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- deferResizing
- editableCollectionView
- editRange
- frozenColumns
- frozenRows
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- layoutDefinition
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- rowsPerItem
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showHeaderCollapseButton
- showMarquee
- showSelectedHeaders
- showSort
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- validateEdits
- viewRange
- virtualizationThreshold

Methods

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel

- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getBindingColumn
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem

- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): MultiRow
```

Initializes a new instance of the **`MultiRow`** class.

In most cases, the **options** parameter will include the value for the **`layoutDefinition`** property.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
MultiRow

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Inherited From
`FlexGrid`
Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Inherited From
`FlexGrid`
Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Inherited From
`FlexGrid`
Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Inherited From
`FlexGrid`
Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Inherited From
`FlexGrid`
Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Inherited From
`FlexGrid`
Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **`itemsSource`**.

The column generation depends on the **`itemsSource`** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **`autoGenerateColumns`** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Inherited From
`FlexGrid`
Type
boolean

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Inherited From
`FlexGrid`
Type
`AutoSizeMode`

bottomLeftCells

Gets the **`GridPanel`** that contains the bottom left cells.

The **`bottomLeftCells`** panel appears below the row headers, to the left of the **`columnFooters`** panel.

Inherited From
`FlexGrid`
Type
`GridPanel`

cellFactory

Gets or sets the **`CellFactory`** that creates and updates cells for this grid.

Inherited From
`FlexGrid`
Type
`CellFactory`

cells

Gets the **`GridPanel`** that contains the data cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

centerHeadersVertically

Gets or sets a value that determines whether the content of cells that span multiple rows should be vertically centered.

Type
boolean

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Inherited From
`FlexGrid`
Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Inherited From
`FlexGrid`
Type
`Size`

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Inherited From
`FlexGrid`
Type
boolean

collapsedHeaders

Gets or sets a value that determines whether column headers should be collapsed and displayed as a single row containing the group headers.

If you set the `collapsedHeaders` property to **true**, remember to set the **header** property of every group in order to avoid empty header cells.

Setting the `collapsedHeaders` property to **null** causes the grid to show all header information (groups and columns). In this case, the first row will show the group headers and the remaining rows will show the individual column headers.

Type
boolean

collectionView

Gets the `ICollectionView` that contains the grid data.

Inherited From
`FlexGrid`
Type
`ICollectionView`

columnFooters

Gets the **GridPanel** that contains the column footer cells.

The **columnFooters** panel appears below the grid cells, to the right of the **bottomLeftCells** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **columnFooters** panel to display summary data for columns that have the **aggregate** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Inherited From
FlexGrid
Type
GridPanel

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Inherited From
FlexGrid
Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Inherited From
`FlexGrid`
Type
string

columns

Gets the grid's column collection.

Inherited From
`FlexGrid`
Type
`ColumnCollection`

controlRect

Gets the bounding rectangle of the control in page coordinates.

Inherited From
`FlexGrid`
Type
`Rect`

STATIC controlTemplate

Gets or sets the template used to instantiate `FlexGrid` controls.

Inherited From
`FlexGrid`
Type
any

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **deferResizing** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Inherited From
FlexGrid
Type
boolean

editableCollectionView

Gets the **IEditableCollectionView** that contains the grid data.

Inherited From
FlexGrid
Type
IEditableCollectionView

editRange

Gets a **CellRange** that identifies the cell currently being edited.

Inherited From
FlexGrid
Type
CellRange

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Inherited From
FlexGrid
Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Inherited From
`FlexGrid`
Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Inherited From
`FlexGrid`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Inherited From
`FlexGrid`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Inherited From
FlexGrid
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** that contains items shown on the grid.

Inherited From
FlexGrid
Type
any

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **getError** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Inherited From
FlexGrid
Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **`MoveDown`**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Inherited From
`FlexGrid`
Type
`KeyAction`

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **`None`**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **`Cycle`**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **`CycleOut`** setting that causes the selection to move through the cells (as **`Cycle`**), and then on to the next/previous control on the page when the last or first cells are selected.

Inherited From
`FlexGrid`
Type
`KeyAction`

layoutDefinition

Gets or sets an array that defines the layout of the rows used to display each data item.

The array contains a list of cell group objects which have the following properties:

- **header**: Group header (shown when the headers are collapsed)
- **colspan**: Number of grid columns spanned by the group
- **cells**: Array of cell objects, which extend **Column** with a **colspan** property.

When the **`layoutDefinition`** property is set, the grid scans the cells in each group as follows:

1. The grid calculates the colspan of the group either as group's own colspan or as span of the widest cell in the group, whichever is wider.
2. If the cell fits the current row within the group, it is added to the current row.
3. If it doesn't fit, it is added to a new row.

When all groups are ready, the grid calculates the number of rows per record to the maximum rowspan of all groups, and adds rows to each group to pad their height as needed.

This scheme is simple and flexible. For example:

```
{ header: 'Group 1', cells: [{ binding: 'c1' }, { bnding: 'c2'}, { binding: 'c3' }]}
```

The group has colspan 1, so there will be one cell per column. The result is:

```
| C1 |
| C2 |
| C3 |
```

To create a group with two columns, set **colspan** property of the group:

```
{ header: 'Group 1', colspan: 2, cells:[{ binding: 'c1' }, { binding: 'c2'}, { binding: 'c3' }]}
```

The cells will wrap as follows:

```
| C1  | C2 |
| C3       |
```

Note that the last cell spans two columns (to fill the group).

You can also specify the colspan on individual cells rather than on the group:

```
{ header: 'Group 1', cells: [{binding: 'c1', colspan: 2 }, { bnding: 'c2'}, { binding: 'c3' }]}
```

Now the first cell has colspan 2, so the result is:

```
| C1       |
| C2 |  C3 |
```

Because cells extend the **Column** class, you can add all the usual **Column** properties to any cells:

```
{ header: 'Group 1', cells: [
   { binding: 'c1', colspan: 2 },
   { bnding: 'c2'},
   { binding: 'c3', format: 'n0', required: false, etc... }
]}
```

Type
any[]

mergeManager

Gets or sets the **MergeManager** object responsible for determining how cells should be merged.

Inherited From
FlexGrid
Type
MergeManager

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **newRowAtTop** property to true, and you want the new row template to remain visible at all times, set the **frozenRows** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **allowAddNew** property is set to true and if the **itemsSource** object supports adding new items.

Inherited From
FlexGrid
Type
boolean

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **`preserveOutlineState`** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Inherited From
`FlexGrid`
Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Inherited From
`FlexGrid`
Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Inherited From
`FlexGrid`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Inherited From
`FlexGrid`
Type
string

rowHeaders

Gets the `GridPanel` that contains the row header cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

rows

Gets the grid's row collection.

Inherited From
`FlexGrid`
Type
`RowCollection`

rowsPerItem

Gets the number of rows used to display each item.

This value is calculated automatically based on the value of the **layoutDefinition** property.

Type
number

scrollPosition

Gets or sets a **`Point`** that represents the value of the grid's scrollbars.

Inherited From
`FlexGrid`
Type
`Point`

scrollSize

Gets the size of the grid content in pixels.

Inherited From
`FlexGrid`
Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when **`selectionMode`** is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when **`selectionMode`** is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selection

Gets or sets the current selection.

Inherited From
`FlexGrid`
Type
`CellRange`

selectionMode

Gets or sets the current selection mode.

Inherited From
`FlexGrid`
Type
`SelectionMode`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Inherited From
`FlexGrid`
Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **`showDropDown`** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`FlexGrid`
Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **`itemValidator`** property or the **`getError`** property on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showHeaderCollapseButton

Gets or sets a value that determines whether the grid should display a button in the column header panel to allow users to collapse and expand the column headers.

If the button is visible, clicking on it will cause the grid to toggle the value of the **collapsedHeaders** property.

Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Inherited From
`FlexGrid`
Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Inherited From
`FlexGrid`
Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Inherited From
`FlexGrid`
Type
boolean

topLeftCells

Gets the **GridPanel** that contains the top left cells (to the left of the column headers).

Inherited From
FlexGrid
Type
GridPanel

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Inherited From
FlexGrid
Type
number

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **getError** method on the grid's **itemsSource**.

Inherited From
FlexGrid
Type
boolean

viewRange

Gets the range of cells currently in view.

Inherited From
FlexGrid
Type
CellRange

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Inherited From
`FlexGrid`
Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.

- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`Control`
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
FlexGrid
Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Inherited From
FlexGrid
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
`Control`
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Inherited From
`FlexGrid`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Inherited From
FlexGrid
Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Inherited From
`FlexGrid`
Returns
void

getBindingColumn

```
getBindingColumn(p: GridPanel, r: number, c: number): Column
```

Gets the **`Column`** object used to bind a data item to a grid cell.

Parameters

- **p**: **GridPanel**
  `GridPanel` that contains the cell.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.

Returns
`Column`

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
`FlexGrid`
Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Inherited From
FlexGrid
Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Returns
`Column`

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Inherited From
FlexGrid
Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.
- **c**: **number**
  Column index of the cell to inspect.

Inherited From
FlexGrid
Returns
SelectedState

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Inherited From
FlexGrid
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Inherited From
FlexGrid
Returns
boolean

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the grid layout and content, or just the content.

Inherited From
FlexGrid
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.

- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.

- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Inherited From
FlexGrid
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.
- **c**: **number**
  Index of the column to scroll into view.

Inherited From
FlexGrid
Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

Parameters

- **rng**: **any**
  Range to select.
- **show**: **any** OPTIONAL
  Whether to scroll the new selection into view.

Inherited From
FlexGrid
Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Sets the value of a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the grid to show the change.

Inherited From
`FlexGrid`
Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Hidden rows and columns are skipped.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Inherited From
FlexGrid
Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Inherited From
`FlexGrid`
Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Inherited From
`FlexGrid`
Arguments
`CellEditEndingEventArgs`

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **allowDelete** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **allowDelete** property).

The event handler may cancel the row deletion.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedColumn

Occurs when the user finishes dragging a column.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedRow

Occurs when the user finishes dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggingRowOver

Occurs as the user drags a row to a new position.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **itemFormatter** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Inherited From
FlexGrid
Arguments
FormatItemEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

resizedColumn

Occurs when the user finishes resizing a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizedRow

Occurs when the user finishes resizing rows.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **`rowEditStarted`** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarted

Occurs after a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

selectionChanged

Occurs after selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

updatedLayout

Occurs after the grid has updated its internal layout.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatingLayout

Occurs before the grid updates its internal layout.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

wijmo.input Module

File
`wijmo.input.js`
Module
wijmo.input

Defines input controls for strings, numbers, dates, times, and colors.

Classes

- AutoComplete
- Calendar
- ColorPicker
- ComboBox
- DropDown
- FormatItemEventArgs
- InputColor
- InputDate
- InputDateTime
- InputMask
- InputNumber
- InputTime
- ListBox
- Menu
- MultiAutoComplete
- MultiSelect
- Popup

Enums

- DateSelectionMode
- PopupTrigger

AutoComplete Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
ComboBox
Derived Classes
MultiAutoComplete, **WjAutoComplete**

The **AutoComplete** control is an input control that allows callers to customize the item list as the user types.

The control is similar to the **ComboBox**, except the item source is a function (**itemsSourceFunction**) rather than a static list. For example, you can look up items on remote databases as the user types.

The example below creates an **AutoComplete** control and populates it using a 'countries' array. The **AutoComplete** searches for the country as the user types, and narrows down the list of countries that match the current input.

Example

Show me (http://jsfiddle.net/Wijmo5/8HnLx)

Constructor

- constructor

Properties

- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- maxDropDownHeight
- maxDropDownWidth
- maxItems
- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- text

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?: any): AutoComplete
```

Initializes a new instance of the **AutoComplete** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: **any** OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
AutoComplete

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

collectionView

Gets the **ICollectionView** object used as the item source.

Inherited From
ComboBox
Type
ICollectionView

STATIC controlTemplate

Gets or sets the template used to instantiate **DropDown** controls.

Inherited From
DropDown
Type
any

cssMatch

Gets or sets the name of the CSS class used to highlight any parts of the content that match the search terms.

Type
string

delay

Gets or sets the delay, in milliseconds, between when a keystroke occurs and when the search is performed.

Type
number

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
ComboBox
Type
string

dropDown

Gets the drop down element shown when the **isDroppedDown** property is set to true.

Inherited From
DropDown
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the `itemsSource` collection.

Inherited From
`ComboBox`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ComboBox
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

itemsSourceFunction

Gets or sets a function that provides list items dynamically as the user types.

The function takes three parameters:

- the query string typed by the user
- the maximum number of items to return
- the callback function to call when the results become available

For example:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // get results from the server
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // return results to the control
    callback(response);
  });
};
```

Type
Function

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
ComboBox
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

maxItems

Gets or sets the maximum number of items to display in the drop-down list.

Type
number

minLength

Gets or sets the minimum input length to trigger auto-complete suggestions.

Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

searchMemberPath

Gets or sets a string containing a comma-separated list of properties to use when searching for items.

By default, the **AutoComplete** control searches for matches against the property specified by the **displayMemberPath** property. The **searchMemberPath** property allows you to search using additional properties.

For example, the code below would cause the control to display the company name and search by company name, symbol, and country:

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

Type
string

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
ComboBox
Type
number

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
ComboBox
Type
any

- **selectedValue**

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

- **selectedValuePath**

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

- **showDropDownButton**

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

- **text**

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
ComboBox
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

Calendar Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
WjCalendar

The **Calendar** control displays a one-month calendar and allows users to select a date.

You may use the **min** and **max** properties to restrict the range of dates that the user can select.

For details about using the **min** and **max** properties, please see the Using the min and max properties topic.

Use the **value** property to get or set the currently selected date.

Use the **selectionMode** property to determine whether users should be allowed to select days, months, or no values at all.

The **Calendar** control supports the following keyboard commands:

Key Combination	**Moves Selection To**
Left	Previous day
Right	Next day
Up	Previous week
Down	Next week
PgUp	Previous month
PgDn	Next month
Alt + PgUp	Previous year
Alt + PgDn	Next year
Home	**min** value (if defined) or first of the month
End	**max** value (if defined) or last of the month
Alt + End	Today's date

The example below shows a **Date** value with date and time information using an **InputDate** and an **InputTime** control. Notice how both controls are bound to the same controller variable, and each edits the appropriate information (either date or time). The example also shows a **Calendar** control that allows users to select the date with a single click.

Example

Show me (http://jsfiddle.net/Wijmo5/vgc3Y)

Constructor

- constructor

Properties

- controlTemplate
- displayMonth
- firstDayOfWeek
- formatDayHeaders
- formatDays
- formatMonths
- formatYear
- formatYearMonth
- hostElement
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- max
- min
- monthView
- rightToLeft
- selectionMode
- showHeader
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onDisplayMonthChanged
- onFormatItem
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- displayMonthChanged
- formatItem
- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): Calendar
```

Initializes a new instance of the **Calendar** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
Calendar

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **Calendar** controls.

Type
any

displayMonth

Gets or sets the month displayed in the calendar.

Type
Date

firstDayOfWeek

Gets or sets a value that represents the first day of the week, the one displayed in the first column of the calendar.

Setting this property to null causes the calendar to use the default for the current culture. In the English culture, the first day of the week is Sunday (0); in most European cultures, the first day of the week is Monday (1).

Type
number

formatDayHeaders

Gets or sets the format used to display the headers above the days in month view.

The default value for this property is 'ddd'.

Type
string

formatDays

Gets or sets the format used to display the days in month view.

The default value for this property is 'd ' (the space after the 'd' prevents the format from being interpreted as 'd', the standard format used to represent the short date pattern).

Type
string

formatMonths

Gets or sets the format used to display the months in year view.

The default value for this property is 'MMM'.

Type
string

formatYear

Gets or sets the format used to display the year above the months in year view.

The default value for this property is 'yyyy'.

Type
string

formatYearMonth

Gets or sets the format used to display the month and year above the calendar in month view.

The default value for this property is 'y'.

Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function to customize dates in the calendar.

The formatter function can add any content to any date. It allows complete customization of the appearance and behavior of the calendar.

If specified, the function takes two parameters:

- the date being formatted
- the HTML element that represents the date

For example, the code below shows weekends with a yellow background:

```
calendar.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

Type
Function

itemValidator

Gets or sets a validator function to determine whether dates are valid for selection.

If specified, the validator function should take one parameter representing the date to be tested, and should return false if the date is invalid and should not be selectable.

For example, the code below shows weekends in a disabled state and prevents users from selecting those dates:

```
calendar.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

Type
Function

max

Gets or sets the latest date that the user can select in the calendar.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Type
Date

min

Gets or sets the earliest date that the user can select in the calendar.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Type
Date

monthView

Gets or sets a value indicating whether the calendar displays a month or a year.

Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectionMode

Gets or sets a value that indicates whether users can select days, months, or no values at all.

Type
`DateSelectionMode`

showHeader

Gets or sets a value indicating whether the control displays the header area with the current month and navigation buttons.

Type
boolean

value

Gets or sets the currently selected date.

Type
Date

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onDisplayMonthChanged

```
onDisplayMonthChanged(e?: EventArgs): void
```

Raises the **displayMonthChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Indicates whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

displayMonthChanged

Occurs after the **displayMonth** property changes.

Arguments
EventArgs

formatItem

Occurs when an element representing a day in the calendar has been created.

This event can be used to format calendar items for display. It is similar in purpose to the **itemFormatter** property, but has the advantage of allowing multiple independent handlers.

For example, the code below uses the **formatItem** event to disable weekends so they appear dimmed in the calendar:

```
// disable Sundays and Saturdays
calendar.formatItem.addHandler(function (s, e) {
  var day = e.data.getDay();
  if (day == 0 || day == 6) {
    wijmo.addClass(e.item, 'wj-state-disabled');
  }
});
```

Arguments
FormatItemEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

lostFocus

Occurs when the control loses the focus.

Inherited From
Control
Arguments
EventArgs

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

ColorPicker Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
WjColorPicker

The **ColorPicker** control allows users to select a color by clicking on panels to adjust color channels (hue, saturation, brightness, alpha).

Use the **value** property to get or set the currently selected color.

The control is used as a drop-down for the **InputColor** control.

Example

Show me (http://jsfiddle.net/Wijmo5/84xvsz90)

Constructor

- constructor

Properties

- controlTemplate
- hostElement
- isDisabled
- isTouching
- isUpdating
- palette
- rightToLeft
- showAlphaChannel
- showColorString
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): ColorPicker
```

Initializes a new instance of the **`ColorPicker`** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
ColorPicker

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **`ColorPicker`** controls.

Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

palette

Gets or sets an array that contains the colors in the palette.

The palette contains ten colors, represented by an array with ten strings. The first two colors are usually white and black.

Type
string[]

- **rightToLeft**

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

- **showAlphaChannel**

Gets or sets a value indicating whether the **`ColorPicker`** allows users to edit the color's alpha channel (transparency).

Type
boolean

- **showColorString**

Gets or sets a value indicating whether the **`ColorPicker`** shows a string representation of the current color.

Type
boolean

- **value**

Gets or sets the currently selected color.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

ComboBox Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
DropDown
Derived Classes
AutoComplete, **InputTime**, **Menu**, **MultiSelect**, **WjComboBox**

The **ComboBox** control allows users to pick strings from lists.

The control automatically completes entries as the user types, and allows users to show a drop-down list with the items available.

Use the **itemsSource** property to populate the list of options. The items may be strings or objects. If the items are objects, use the **displayMemberPath** to define which property of the items will be displayed in the list and use the **selectedValuePath** property to define which property of the items will be used to set the combo's **selectedValue** property.

Use the **selectedIndex** or the **text** properties to determine which item is currently selected.

The **isEditable** property determines whether users can enter values that are not present in the list.

The example below creates a **ComboBox** control and populates it with a list of countries. The **ComboBox** searches for the country as the user types. The **isEditable** property is set to false, so the user is forced to select one of the items in the list.

The example also shows how to create and populate a **ComboBox** using an HTML **<select>** element with **<option>** child elements.

Example

Show me (http://jsfiddle.net/Wijmo5/8HnLx)

Constructor

- constructor

Properties

- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- maxDropDownHeight
- maxDropDownWidth
- placeholder
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- text

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): ComboBox
```

Initializes a new instance of the **ComboBox** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
ComboBox

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

collectionView

Gets the **ICollectionView** object used as the item source.

Type
ICollectionView

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Type
`Event`

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Type
ListBox

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Type
number

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Type
any

- **selectedValue**

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Type
any

- **selectedValuePath**

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Type
string

- **showDropDownButton**

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

- **text**

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

DropDown Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
ComboBox, **InputColor**, **InputDate**

DropDown control (abstract).

Contains an input element and a button used to show or hide the drop-down.

Derived classes must override the _createDropDown method to create whatever editor they want to show in the drop down area (a list of items, a calendar, a color editor, etc).

Constructor

- constructor

Properties

- autoExpandSelection
- controlTemplate
- dropDown
- dropDownCssClass
- hostElement
- inputElement
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- placeholder
- rightToLeft
- showDropDownButton
- text

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged

Constructor

constructor

```
constructor(element: any, options?): DropDown
```

Initializes a new instance of the **DropDown** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
DropDown

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Type
boolean

STATIC controlTemplate

Gets or sets the template used to instantiate **DropDown** controls.

Type
any

dropDown

Gets the drop down element shown when the **isDroppedDown** property is set to true.

Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

- **isUpdating**

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

- **placeholder**

Gets or sets the string shown as a hint when the control is empty.

Type
string

- **rightToLeft**

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

- **showDropDownButton**

Gets or sets a value that indicates whether the control should display a drop-down button.

Type
boolean

- **text**

Gets or sets the text shown on the control.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Arguments
`EventArgs`

FormatItemEventArgs Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
EventArgs

Provides arguments for the **formatItem** event.

Constructor

- constructor

Properties

- data
- empty
- index
- item

Constructor

constructor

```
constructor(index: number, data: any, item: HTMLElement): FormatItemEventArgs
```

Initializes a new instance of the **`FormatItemEventArgs`** class.

Parameters

- **index**: **number**
  Index of the item being formatted.
- **data**: **any**
  Data item being formatted.
- **item**: **HTMLElement**
  Element that represents the list item to be formatted.

Returns
FormatItemEventArgs

Properties

data

Gets the data item being formatted.

Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

index

Gets the index of the data item in the list.

Type
number

item

Gets a reference to the element that represents the list item to be formatted.

Type
HTMLElement

InputColor Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
DropDown
Derived Classes
WjInputColor

The **InputColor** control allows users to select colors by typing in HTML-supported color strings, or to pick colors from a drop-down that shows a **ColorPicker** control.

Use the **value** property to get or set the currently selected color.

Example

Show me (http://jsfiddle.net/Wijmo5/84xvsz90)

Constructor

- constructor

Properties

- autoExpandSelection
- colorPicker
- controlTemplate
- dropDown
- dropDownCssClass
- hostElement
- inputElement
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- palette
- placeholder
- rightToLeft
- showAlphaChannel
- showDropDownButton
- text
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputColor
```

Initializes a new instance of the **InputColor** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
InputColor

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

colorPicker

Gets a reference to the **ColorPicker** control shown in the drop-down.

Type
ColorPicker

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

palette

Gets or sets an array that contains the colors in the palette.

The palette contains ten colors, represented by an array with ten strings. The first two colors are usually white and black.

Type
string[]

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showAlphaChannel

Gets or sets a value indicating whether the **`ColorPicker`** allows users to edit the color's alpha channel (transparency).

Type
boolean

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Type
string

value

Gets or sets the current color.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

InputDate Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
DropDown
Derived Classes
InputDateTime, **WjInputDate**

The **InputDate** control allows users to type in dates using any format supported by the **Globalize** class, or to pick dates from a drop-down box that shows a **Calendar** control.

Use the **min** and **max** properties to restrict the range of values that the user can enter.

For details about using the **min** and **max** properties, please see the Using the min and max properties topic.

Use the **value** property to gets or set the currently selected date.

The example below shows a **Date** value (that includes date and time information) using an **InputDate** and an an **InputTime** control. Notice how both controls are bound to the same controller variable, and each edits the appropriate information (either date or time). The example also shows a **Calendar** control that you can use to select the date with a single click.

Example

Show me (http://jsfiddle.net/Wijmo5/vgc3Y)

Constructor

- constructor

Properties

- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- hostElement
- inputElement
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- mask
- max
- min
- placeholder
- rightToLeft
- selectionMode
- showDropDownButton
- text
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputDate
```

Initializes a new instance of the **InputDate** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
InputDate

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

calendar

Gets a reference to the **Calendar** control shown in the drop-down box.

Type
Calendar

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

format

Gets or sets the format used to display the selected date.

The format string is expressed as a .NET-style **`Date format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Type
HTMLInputElement

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Type
string

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function to customize dates in the drop-down calendar.

The formatter function can add any content to any date. It allows complete customization of the appearance and behavior of the calendar.

If specified, the function takes two parameters:

- the date being formatted
- the HTML element that represents the date

For example, the code below shows weekends with a yellow background:

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

Type
Function

itemValidator

Gets or sets a validator function to determine whether dates are valid for selection.

If specified, the validator function should take one parameter representing the date to be tested, and should return false if the date is invalid and should not be selectable.

For example, the code below prevents users from selecting dates that fall on weekends:

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

Type
Function

mask

Gets or sets a mask to use while editing.

The mask format is the same one that the **InputMask** control uses.

If specified, the mask must be compatible with the value of the **format** property. For example, the mask '99/99/9999' can be used for entering dates formatted as 'MM/dd/yyyy'.

Type
string

max

Gets or sets the latest date that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Type
Date

min

Gets or sets the earliest date that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Type
Date

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
DropDown
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
Control
Type
boolean

selectionMode

Gets or sets a value that indicates whether users can select days, months, or no values at all.

This property affects the behavior of the drop-down calendar, but not the format used to display dates. If you set **selectionMode** to 'Month', you should normally set the **format** property to 'MMM yyyy' or some format that does not include the day. For example:

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

Type
DateSelectionMode

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Type
string

value

Gets or sets the current date.

Type
Date

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

InputDateTime Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
InputDate
Derived Classes
WjInputDateTime

The **InputDateTime** control extends the **InputDate** control to allows users to input dates and times, either by typing complete date/time values in any format supported by the **Globalize** class, or by picking dates from a drop-down calendar and times from a drop-down list.

Use the **min** and **max** properties to restrict the range of dates that the user can enter.

Use the **timeMin** and **timeMax** properties to restrict the range of times that the user can enter.

Use the **value** property to gets or set the currently selected date/time.

Constructor

- constructor

Properties

- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- hostElement
- inputElement
- inputTime
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- mask
- max
- min
- placeholder
- rightToLeft
- selectionMode
- showDropDownButton
- text
- timeFormat
- timeMax
- timeMin
- timeStep
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputDateTime
```

Initializes a new instance of the **InputDateTime** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
InputDateTime

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

calendar

Gets a reference to the **Calendar** control shown in the drop-down box.

Inherited From
InputDate
Type
Calendar

STATIC controlTemplate

Gets or sets the template used to instantiate **`InputDateTime`** controls.

Type
any

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

format

Gets or sets the format used to display the selected date.

The format string is expressed as a .NET-style **`Date format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Inherited From
`InputDate`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`InputDate`
Type
HTMLInputElement

inputTime

Gets a reference to the inner **`InputTime`** control so you can access its full object model.

Type
`InputTime`

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Inherited From
`InputDate`
Type
string

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function to customize dates in the drop-down calendar.

The formatter function can add any content to any date. It allows complete customization of the appearance and behavior of the calendar.

If specified, the function takes two parameters:

- the date being formatted
- the HTML element that represents the date

For example, the code below shows weekends with a yellow background:

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

Inherited From
`InputDate`
Type
Function

itemValidator

Gets or sets a validator function to determine whether dates are valid for selection.

If specified, the validator function should take one parameter representing the date to be tested, and should return false if the date is invalid and should not be selectable.

For example, the code below prevents users from selecting dates that fall on weekends:

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

Inherited From
`InputDate`
Type
Function

mask

Gets or sets a mask to use while editing.

The mask format is the same one that the **`InputMask`** control uses.

If specified, the mask must be compatible with the value of the **`format`** property. For example, the mask '99/99/9999' can be used for entering dates formatted as 'MM/dd/yyyy'.

Inherited From
`InputDate`
Type
string

max

Gets or sets the latest date that the user can enter.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`InputDate`
Type
Date

min

Gets or sets the earliest date that the user can enter.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`InputDate`
Type
Date

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectionMode

Gets or sets a value that indicates whether users can select days, months, or no values at all.

This property affects the behavior of the drop-down calendar, but not the format used to display dates. If you set **`selectionMode`** to 'Month', you should normally set the **`format`** property to 'MMM yyyy' or some format that does not include the day. For example:

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

Inherited From
`InputDate`
Type
`DateSelectionMode`

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`InputDate`
Type
string

timeFormat

Gets or sets the format used to display times in the drop-down list.

This property does not affect the value shown in the control's input element. That value is formatted using the **`format`** property.

The format string is expressed as a .NET-style **`time format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Type
string

timeMax

Gets or sets the latest time that the user can enter.

Type
Date

timeMin

Gets or sets the earliest time that the user can enter.

Type
Date

timeStep

Gets or sets the number of minutes between entries in the drop-down list of times.

Type
number

value

Gets or sets the current date.

Inherited From
`InputDate`
Type
Date

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputDate
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputDate`
Arguments
`EventArgs`

InputMask Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
WjInputMask

The **InputMask** control provides a way to govern what a user is allowed to input.

The control prevents users from accidentally entering invalid data and saves time by skipping over literals (such as slashes in dates) as the user types.

The mask used to validate the input is defined by the **mask** property, which may contain one or more of the following special characters:

0
Digit.
9
Digit or space.
#
Digit, sign, or space.
L
Letter.
l
Letter or space.
A
Alphanumeric.
a
Alphanumeric or space.
.
Localized decimal point.
,
Localized thousand separator.
:
Localized time separator.
/
Localized date separator.
$
Localized currency symbol.
<
Converts characters that follow to lowercase.
>
Converts characters that follow to uppercase.
|
Disables case conversion.
****
Escapes any character, turning it into a literal.

9
DBCS Digit.
J
DBCS Hiragana.
G
DBCS big Hiragana.
K
DBCS Katakana.
N
DBCS big Katakana.
K
SBCS Katakana.
N
SBCS big Katakana.
Z
Any DBCS character.
H
Any SBCS character.
All others
Literals.

Constructor

- constructor

Properties

- controlTemplate
- hostElement
- inputElement
- isDisabled
- isRequired
- isTouching
- isUpdating
- mask
- maskFull
- placeholder
- promptChar
- rawValue
- rightToLeft
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputMask
```

Initializes a new instance of the **InputMask** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
InputMask

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **InputMask** controls.

Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Type
HTMLInputElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isRequired

Gets or sets a value indicating whether the control value must be a non-empty string.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

mask

Gets or sets the mask used to validate the input as the user types.

The mask is defined as a string with one or more of the masking characters listed in the **`InputMask`** topic.

Type
string

maskFull

Gets a value that indicates whether the mask has been completely filled.

Type
boolean

placeholder

Gets or sets the string shown as a hint when the control is empty.

Type
string

promptChar

Gets or sets the symbol used to show input positions in the control.

Type
string

rawValue

Gets or sets the raw value of the control (excluding mask literals).

The raw value of the control excludes prompt and literal characters. For example, if the **`mask`** property is set to "AA-9999" and the user enters the value "AB-1234", the **`rawValue`** property will return "AB1234", excluding the hyphen that is part of the mask.

Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

value

Gets or sets the text currently shown in the control.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

InputNumber Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
WjInputNumber

The **InputNumber** control allows users to enter numbers.

The control prevents users from accidentally entering invalid data and formats the number as it is edited.

Pressing the minus key reverses the sign of the value being edited, regardless of cursor position.

You may use the **min** and **max** properties to limit the range of acceptable values, and the **step** property to provide spinner buttons that increase or decrease the value with a click.

For details about using the **min** and **max** properties, please see the Using the min and max properties topic.

Use the **value** property to get or set the currently selected number.

The example below creates several **InputNumber** controls and shows the effect of using different formats, ranges, and step values.

Example

Show me (http://jsfiddle.net/Wijmo5/Cf9L9)

Constructor

- constructor

Properties

- controlTemplate
- format
- hostElement
- inputElement
- inputType
- isDisabled
- isReadOnly
- isRequired
- isTouching
- isUpdating
- max
- min
- placeholder
- rightToLeft
- showSpinner
- step
- text
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputNumber
```

Initializes a new instance of the **InputNumber** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
InputNumber

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **InputNumber** controls.

Type
any

format

Gets or sets the format used to display the number being edited (see **Globalize**).

The format string is expressed as a .NET-style **standard numeric format string (http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx)**.

Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Type
HTMLInputElement

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Type
string

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Type
boolean

isRequired

Gets or sets a value indicating whether the control value must be a number or whether it can be set to null (by deleting the content of the control).

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

max

Gets or sets the largest number that the user can enter.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Type
number

min

Gets or sets the smallest number that the user can enter.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showSpinner

Gets or sets a value indicating whether the control displays spinner buttons to increment or decrement the value (the step property must be set to a value other than zero).

Type
boolean

step

Gets or sets the amount to add or subtract to the **`value`** property when the user clicks the spinner buttons.

Type
number

text

Gets or sets the text shown in the control.

Type
string

value

Gets or sets the current value of the control.

Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
`Control`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

InputTime Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
ComboBox
Derived Classes
WjInputTime

The **InputTime** control allows users to enter times using any format supported by the **Globalize** class, or to pick times from a drop-down list.

The **min**, **max**, and **step** properties determine the values shown in the list.

For details about using the **min** and **max** properties, please see the Using the min and max properties topic.

The **value** property gets or sets a **Date** object that represents the time selected by the user.

The example below shows a **Date** value (that includes date and time information) using an **InputDate** and an **InputTime** control. Notice how both controls are bound to the same controller variable, and each edits the appropriate information (either date or time). The example also shows a **Calendar** control that can be used to select the date with a single click.

Example

Show me (http://jsfiddle.net/Wijmo5/vgc3Y)

Constructor

- constructor

Properties

- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- format
- formatItem
- headerPath
- hostElement
- inputElement
- inputType
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- mask
- max
- maxDropDownHeight
- maxDropDownWidth
- min
- placeholder
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- step
- text
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputTime
```

Initializes a new instance of the **InputTime** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
InputTime

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

collectionView

Gets the **ICollectionView** object used as the item source.

Inherited From
ComboBox
Type
ICollectionView

STATIC controlTemplate

Gets or sets the template used to instantiate **DropDown** controls.

Inherited From
DropDown
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
ComboBox
Type
string

dropDown

Gets the drop down element shown when the **isDroppedDown** property is set to true.

Inherited From
DropDown
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
DropDown
Type
string

format

Gets or sets the format used to display the selected time (see **`Globalize`**).

The format string is expressed as a .NET-style **`time format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Type
HTMLInputElement

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Type
string

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the `itemsSource` collection.

Inherited From
`ComboBox`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ComboBox
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

mask

Gets or sets a mask to use while the user is editing.

The mask format is the same used by the **`InputMask`** control.

If specified, the mask must be compatible with the value of the **`format`** property. For example, you can use the mask '99:99 >LL' for entering short times (format 't').

Type
string

max

Gets or sets the latest time that the user can enter.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Type
Date

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

min

Gets or sets the earliest time that the user can enter.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Type
Date

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

step

Gets or sets the number of minutes between entries in the drop-down list.

The default value for this property is 15 minutes. Setting it to null, zero, or any negative value disables the drop-down.

Type
number

text

Gets or sets the text shown in the control.

Type
string

value

Gets or sets the current input time.

Type
Date

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
ComboBox
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Arguments
`EventArgs`

ListBox Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
WjListBox

The **ListBox** control displays a list of items which may contain plain text or HTML, and allows users to select items with the mouse or the keyboard.

Use the **selectedIndex** property to determine which item is currently selected.

You can populate a **ListBox** using an array of strings or you can use an array of objects, in which case the **displayMemberPath** property determines which object property is displayed on the list.

To display items that contain HTML rather than plain text, set the **isContentHtml** property to true.

The example below creates a **ListBox** control and populates it using a 'countries' array. The control updates its **selectedIndex** and **selectedItem** properties as the user moves the selection.

Example

Show me (http://jsfiddle.net/Wijmo5/8HnLx)

Constructor

- constructor

Properties

- checkedItems
- checkedMemberPath
- collectionView
- displayMemberPath
- hostElement
- isContentHtml
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemRole
- itemsSource
- maxHeight
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getDisplayValue
- getItemChecked
- getTemplate
- initialize
- invalidate
- invalidateAll
- isItemEnabled
- onCheckedItemsChanged
- onFormatItem
- onGotFocus
- onItemChecked
- onItemsChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onSelectedIndexChanged
- refresh
- refreshAll
- removeEventListener
- setItemChecked
- showSelection
- toggleItemChecked

Events

- checkedItemsChanged
- formatItem
- gotFocus
- itemChecked
- itemsChanged
- loadedItems
- loadingItems
- lostFocus
- selectedIndexChanged

Constructor

constructor

```
constructor(element: any, options?): ListBox
```

Initializes a new instance of the **ListBox** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
ListBox

Properties

checkedItems

Gets or sets an array containing the items that are currently checked.

Type
any[]

checkedMemberPath

Gets or sets the name of the property used to control the CheckBoxes placed next to each item.

Use this property to create multi-select LisBoxes. When an item is checked or unchecked, the control raises the **itemChecked** event. Use the **selectedItem** property to retrieve the item that was checked or unchecked, or use the **checkedItems** property to retrieve the list of items that are currently checked.

Type

collectionView

Gets the **ICollectionView** object used as the item source.

Type
ICollectionView

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isContentHtml

Gets or sets a value indicating whether items contain plain text or HTML.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown on the list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
listBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Type
Function

itemRole

Gets or sets the value or the "role" attribute added to the list items. The default value for this property is "option".

Type
string

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the list items.

Type
any

maxHeight

Gets or sets the maximum height of the list.

Type
number

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item.

Type
number

selectedItem

Gets or sets the item that is currently selected.

Type
any

selectedValue

Gets or sets the value of the **`selectedItem`** obtained using the **`selectedValuePath`**.

Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index: number): string
```

Gets the text displayed for the item at a given index (as plain text).

Parameters

- **index**: **number**
  The index of the item.

Returns
string

getDisplayValue

```
getDisplayValue(index: number): string
```

Gets the string displayed for the item at a given index.

The string may be plain text or HTML, depending on the setting of the **`isContentHtml`** property.

Parameters

- **index**: **number**
  The index of the item.

Returns
string

getItemChecked

```
getItemChecked(index: number): boolean
```

Gets the checked state of an item on the list.

This method is applicable only on multi-select ListBoxes (see the **`checkedMemberPath`** property).

Parameters

- **index**: **number**
  Item index.

Returns
boolean

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isItemEnabled

```
isItemEnabled(index: number): void
```

Gets a value that determines whether the item at a given index is enabled.

Parameters

- **index**: **number**
  The index of the item.

Returns
void

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

Raises the **checkedItemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onItemChecked

```
onItemChecked(e?: EventArgs): void
```

Raises the **itemChecked** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onItemsChanged

```
onItemsChanged(e?: EventArgs): void
```

Raises the **itemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

Raises the **`loadedItems`** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadingItems

```
onLoadingItems(e?: EventArgs): void
```

Raises the **`loadingItems`** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **`lostFocus`** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`Control`
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(): void
```

Refreshes the list.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

setItemChecked

```
setItemChecked(index: number, checked: boolean): void
```

Sets the checked state of an item on the list.

This method is applicable only on multi-select ListBoxes (see the **`checkedMemberPath`** property).

Parameters

- **index**: **number**
  Item index.
- **checked**: **boolean**
  Item's new checked state.

Returns
void

showSelection

```
showSelection(): void
```

Highlights the selected item and scrolls it into view.

Returns
void

toggleItemChecked

```
toggleItemChecked(index: number): void
```

Toggles the checked state of an item on the list. This method is applicable only to multi-select ListBoxes (see the **checkedMemberPath** property).

Parameters

- **index**: **number**
  Item index.

Returns
void

Events

checkedItemsChanged

Occurs when the value of the **checkedItems** property changes.

Arguments
EventArgs

formatItem

Occurs when an element representing a list item has been created.

This event can be used to format list items for display. It is similar in purpose to the **itemFormatter** property, but has the advantage of allowing multiple independent handlers.

Arguments
FormatItemEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

itemChecked

Occurs when the current item is checked or unchecked by the user.

This event is raised when the **`checkedMemberPath`** is set to the name of a property to add CheckBoxes to each item in the control.

Use the **`selectedItem`** property to retrieve the item that was checked or unchecked.

Arguments
`EventArgs`

itemsChanged

Occurs when the list of items changes.

Arguments
`EventArgs`

loadedItems

Occurs after the list items have been generated.

Arguments
`EventArgs`

loadingItems

Occurs before the list items are generated.

Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Arguments
`EventArgs`

Menu Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
ComboBox
Derived Classes
WjMenu

The **Menu** control shows a text element with a drop-down list of commands that the user can invoke by click or touch.

The **Menu** control inherits from **ComboBox**, so you populate and style it in the same way that you do the **ComboBox** (see the **itemsSource** property).

The **Menu** control adds an **itemClicked** event that fires when the user selects an item from the menu. The event handler can inspect the **Menu** control to determine which item was clicked. For example:

```
var menu = new wijmo.input.Menu(hostElement);
menu.header = 'Main Menu';
menu.itemsSource = ['option 1', 'option 2', 'option 3'];
menu.itemClicked.addHandler(function(sender, args) {
var menu = sender;
  alert('Thanks for selecting item ' + menu.selectedIndex + ' from menu ' + menu.header + '!');
});
```

The example below illustrates how you can create value pickers, command-based menus, and menus that respond to the **itemClicked** event. The menus in this example are based on HTML **<select;>** and **<option;>** elements.

Example

Show me (http://jsfiddle.net/Wijmo5/BX853)

Constructor

- constructor

Properties

- autoExpandSelection
- collectionView
- command
- commandParameterPath
- commandPath
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- header
- headerPath
- hostElement
- inputElement
- isAnimated
- isButton
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- maxDropDownHeight
- maxDropDownWidth
- owner
- placeholder
- rightToLeft
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- text

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- hide
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemClicked
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll
- show

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemClicked
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): Menu
```

Initializes a new instance of the **Menu** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
Menu

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

collectionView

Gets the **ICollectionView** object used as the item source.

Inherited From
ComboBox
Type
ICollectionView

command

Gets or sets the command to execute when an item is clicked.

Commands are objects that implement two methods:

- **executeCommand(parameter)** This method executes the command.
- **canExecuteCommand(parameter)** This method returns a Boolean value that determines whether the controller can execute the command. If this method returns false, the menu option is disabled.

You can also set commands on individual items using the **`commandPath`** property.

Type
any

commandParameterPath

Gets or sets the name of the property that contains a parameter to use with the command specified by the **`commandPath`** property.

Type
string

commandPath

Gets or sets the name of the property that contains the command to execute when the user clicks an item.

Commands are objects that implement two methods:

- **executeCommand(parameter)** This method executes the command.
- **canExecuteCommand(parameter)** This method returns a Boolean value that determines whether the controller can execute the command. If this method returns false, the menu option is disabled.

Type
string

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

header

Gets or sets the HTML text shown in the `Menu` element.

Type
string

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isButton

Gets or sets a value that determines whether this **`Menu`** should act as a split button instead of a regular menu.

The difference between regular menus and split buttons is what happens when the user clicks the menu header. In regular menus, clicking the header shows or hides the menu options. In split buttons, clicking the header raises the **`itemClicked`** event and/or invokes the command associated with the last option selected by the user as if the user had picked the item from the drop-down list.

If you want to differentiate between clicks on menu items and the button part of a split button, check the value of the **`isDroppedDown`** property of the event sender. If that is true, then a menu item was clicked; if it is false, then the button was clicked.

For example, the code below implements a split button that uses the drop-down list only to change the default item/command, and triggers actions only when the button is clicked:

```
<-- view -->
<wj-menu is-button="true" header="Run" value="browser"
  item-clicked="itemClicked(s, e)">
  <wj-menu-item value="'Internet Explorer'">Internet Explorer</wj-menu-item>
  <wj-menu-item value="'Chrome'">Chrome</wj-menu-item>
  <wj-menu-item value="'Firefox'">Firefox</wj-menu-item>
  <wj-menu-item value="'Safari'">Safari</wj-menu-item>
  <wj-menu-item value="'Opera'">Opera</wj-menu-item>
</wj-menu>

// controller
$scope.browser = 'Internet Explorer';
$scope.itemClicked = function (s, e) {

  // if not dropped down, click was on the button
  if (!s.isDroppedDown) {
    alert('running ' + $scope.browser);
  }
}
```

Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ComboBox
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

owner

Gets or sets the element that owns this **`Menu`**.

This variable is set by the wj-context-menu directive in case a single menu is used as a context menu for several different elements.

Type
HTMLElement

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

- **selectedValuePath**

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

- **showDropDownButton**

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

- **text**

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
ComboBox
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hide

```
hide(): void
```

Hides the menu.

This method is useful if you want to hide a context menu displayed with the **show** method.

Returns
void

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.

- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

Raises the **itemClicked** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

show

```
show(position?: any): void
```

Shows the menu at a given location.

This method is useful if you want to use the menu as a context menu, attached to one or more elements on the page. For example:

```
// create menu
var div = document.createElement('div');
var menu = new wijmo.input.Menu(div, {
    itemsSource: 'New,Open,Save,Exit'.split(','),
    itemClicked: function (s, e) {
        alert('thanks for picking ' + menu.selectedIndex);
    }
});

// use it as a context menu for one or more elements
var element = document.getElementById('btn');
element.addEventListener('contextmenu', function (e) {
    e.preventDefault();
    menu.show(e);
});
```

Parameters

- **position**: **any** OPTIONAL
  An optional **MouseEvent** or reference element that determines the position where the menu should be displayed. If not provided, the menu is displayed at the center of the screen.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemClicked

Occurs when the user picks an item from the menu.

The handler can determine which item was picked by reading the event sender's **`selectedIndex`** property.

Arguments
`EventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

MultiAutoComplete Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
AutoComplete
Derived Classes
WjMultiAutoComplete

The **MultiAutoComplete** control allows users to pick items from lists that contain custom objects or simple strings.

Constructor

- constructor

Properties

- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- maxDropDownHeight
- maxDropDownWidth
- maxItems
- maxSelectedItems
- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedItem
- selectedItems
- selectedMemberPath
- selectedValue
- selectedValuePath
- showDropDownButton
- text

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onSelectedItemsChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged

- selectedItemsChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): MultiAutoComplete
```

Initializes a new instance of the **MultiAutoComplete** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
MultiAutoComplete

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

collectionView

Gets the **ICollectionView** object used as the item source.

Inherited From
ComboBox
Type
ICollectionView

STATIC controlTemplate

Gets or sets the template used to instantiate **DropDown** controls.

Inherited From
DropDown
Type
any

cssMatch

Gets or sets the name of the CSS class used to highlight any parts of the content that match the search terms.

Inherited From
AutoComplete
Type
string

delay

Gets or sets the delay, in milliseconds, between when a keystroke occurs and when the search is performed.

Inherited From
AutoComplete
Type
number

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
ComboBox
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ComboBox
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

itemsSourceFunction

Gets or sets a function that provides list items dynamically as the user types.

The function takes three parameters:

- the query string typed by the user
- the maximum number of items to return
- the callback function to call when the results become available

For example:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // get results from the server
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // return results to the control
    callback(response);
  });
};
```

Inherited From
`AutoComplete`
Type
Function

listBox

Gets the **`ListBox`** control shown in the drop-down.

Inherited From
`ComboBox`
Type
`ListBox`

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

maxItems

Gets or sets the maximum number of items to display in the drop-down list.

Inherited From
`AutoComplete`
Type
number

maxSelectedItems

Gets or sets the maximum number of items that can be selected.

Setting this property to null (the default value) allows users to pick any number of items.

Type
number

minLength

Gets or sets the minimum input length to trigger auto-complete suggestions.

Inherited From
`AutoComplete`
Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

searchMemberPath

Gets or sets a string containing a comma-separated list of properties to use when searching for items.

By default, the **`AutoComplete`** control searches for matches against the property specified by the **`displayMemberPath`** property. The **`searchMemberPath`** property allows you to search using additional properties.

For example, the code below would cause the control to display the company name and search by company name, symbol, and country:

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

Inherited From
`AutoComplete`
Type
string

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
ComboBox
Type
number

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
ComboBox
Type
any

selectedItems

Gets or sets an array containing the items that are currently selected.

Type
any[]

selectedMemberPath

Gets or sets the name of the property used to control which item will be selected.

Type
string

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
ComboBox
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Override the value for indicating control should not display a drop-down button.

Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
ComboBox
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onSelectedItemsChanged

```
onSelectedItemsChanged(e?: EventArgs): void
```

Raises the **selectedItemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

selectedItemsChanged

Occurs when the value of the **`selectedItems`** property changes.

Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

MultiSelect Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
ComboBox
Derived Classes
WjMultiSelect

The **MultiSelect** control allows users to select multiple items from drop-down lists that contain custom objects or simple strings.

The **MultiSelect** control extends **ComboBox**, with all the usual properties, including **itemsSource** and **displayMemberPath**.

Like the **ListBox** control, it has a **checkedMemberPath** property that defines the name of the property that determines whether an item is checked or not.

The items currently checked (selected) can be obtained using the **checkedItems** property.

The control header is fully customizable. By default, it shows up to two items selected and the item count after that. You can change the maximum number of items to display (**maxHeaderItems**), the message shown when no items are selected (**placeholder**), and the format string used to show the item count (**headerFormat**).

Alternatively, you can provide a function to generate the header content based on whatever criteria your application requires (**headerFormatter**).

Constructor

- constructor

Properties

- autoExpandSelection
- checkedItems
- checkedMemberPath
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- headerFormat
- headerFormatter
- headerPath
- hostElement
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isEditable
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- maxDropDownHeight
- maxDropDownWidth
- maxHeaderItems
- placeholder
- rightToLeft
- selectAllLabel
- selectedIndex
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showSelectAllCheckbox
- text

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onCheckedItemsChanged
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- checkedItemsChanged
- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): MultiSelect
```

Initializes a new instance of the **MultiSelect** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
MultiSelect

Properties

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

checkedItems

Gets or sets an array containing the items that are currently checked.

Type
any[]

checkedMemberPath

Gets or sets the name of the property used to control the checkboxes placed next to each item.

Type
string

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

headerFormat

Gets or sets the format string used to create the header content when the control has more than **`maxHeaderItems`** items checked.

The format string may contain the '{count}' replacement string which gets replaced with the number of items currently checked. The default value for this property in the English culture is '{count:n0} items selected'.

Type
string

headerFormatter

Gets or sets a function that gets the HTML in the control header.

By default, the control header content is determined based on the **`placeholder`**, **`maxHeaderItems`**, and on the current selection.

You may customize the header content by specifying a function that returns a custom string based on whatever criteria your application requires.

Type
Function

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

- **isDroppedDown**

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

- **isEditable**

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

- **isReadOnly**

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

- **isRequired**

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
`ComboBox`
Type
Function

itemsSource

Gets or sets the array or `ICollectionView` object that contains the items to select from.

Inherited From
`ComboBox`
Type
any

listBox

Gets the `ListBox` control shown in the drop-down.

Inherited From
`ComboBox`
Type
`ListBox`

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

maxHeaderItems

Gets or sets the maximum number of items to display on the control header.

If no items are selected, the header displays the text specified by the **`placeholder`** property.

If the number of selected items is smaller than or equal to the value of the **`maxHeaderItems`** property, the selected items are shown in the header.

If the number of selected items is greater than **`maxHeaderItems`**, the header displays the selected item count instead.

Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectAllLabel

Gets or sets the string to be used as a label for the "Select All" checkbox that is displayed when the **`showSelectAllCheckbox`** property is set to true.

This property is set to null by default, which causes the control to show a localized version of the string "Select All".

Type
string

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

- **showDropDownButton**

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

- **showSelectAllCheckbox**

Gets or sets whether the control should display a "Select All" checkbox above the items to select or de-select all items.

Type
boolean

- **text**

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
ComboBox
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

Raises the **checkedItemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

checkedItemsChanged

Occurs when the value of the **`checkedItems`** property changes.

Arguments
`EventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

Popup Class

File
`wijmo.input.js`
Module
wijmo.input
Base Class
Control
Derived Classes
WjPopup

Class that shows an element as a popup.

Popups may be have **owner** elements, in which case they behave as rich tooltips that may be shown or hidden based on actions specified by the **showTrigger** and **hideTrigger** properties.

Popups with no owner elements behave like dialogs. They are centered on the screen and displayed using the **show** method.

To close a **Popup**, call the **hide** method.

Alternatively, any clickable elements within a **Popup** that have the classes starting with the 'wj-hide' string will hide the **Popup** when clicked and will set the **dialogResult** property to the class name so the caller may take appropriate action.

For example, the **Popup** below will be hidden when the user presses the OK or Cancel buttons, and the **dialogResult** property will be set to either 'wj-hide-cancel' or 'wj-hide-ok':

```
<button id="btnPopup">Show Popup</button>
<wj-popup owner="#btnPopup" style="padding:12px">
  <p>Press one of the buttons below to hide the Popup.</p>
  <hr/>
  <button class="wj-hide-ok" ng-click="handleOK()">OK</button>
  <button class="wj-hide-cancel">Cancel</button>
</wj-popup>
```

Constructor

- constructor

Properties

- content
- dialogResult
- dialogResultEnter
- fadeIn
- fadeOut
- hideTrigger
- hostElement
- isDisabled
- isTouching
- isUpdating
- isVisible
- modal
- owner
- removeOnHide
- rightToLeft
- showTrigger

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hide
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onHidden
- onHiding
- onLostFocus
- onShowing
- onShown
- refresh
- refreshAll
- removeEventListener
- show

Events

- gotFocus
- hidden
- hiding
- lostFocus
- showing
- shown

Constructor

constructor

```
constructor(element: any, options?: any): Popup
```

Initializes a new instance of the **Popup** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
Popup

Properties

content

Gets or sets the HTML element contained in this **Popup**.

Type
HTMLElement

dialogResult

Gets or sets a value that can be used for handling the content of the **Popup** after it is hidden.

This property is set to null when the **Popup** is displayed, and it can be set in response to button click events or in the call to the **hide** method.

Type
any

dialogResultEnter

Gets or sets a value to be used as a **dialogResult** when the user presses the Enter key while the **Popup** is visible.

If the user presses Enter and the **dialogResultEnter** property is not null, the popup checks whether all its child elements are in a valid state. If so, the popup is closed and the **dialogResult** property is set to the value of the **dialogResultEnter** property.

Type
any

fadeIn

Gets or sets a value that determines whether the **Popup** should use a fade-out animation when it is shown.

Type
boolean

fadeOut

Gets or sets a value that determines whether the **Popup** should use a fade-out animation when it is hidden.

Type
boolean

hideTrigger

Gets or sets the actions that hide the **Popup**.

By default, the **hideTrigger** property is set to **Blur**, which hides the popup when it loses focus.

If you set the **hideTrigger** property to **Click**, the popup will be hidden only when the owner element is clicked.

If you set the **hideTrigger** property to **None**, the popup will be hidden only when the **hide** method is called.

Type
PopupTrigger

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

isVisible

Gets a value that determines whether the **Popup** is currently visible.

Type
boolean

modal

Gets or sets a value that determines whether the **Popup** should be displayed as a modal dialog.

Modal dialogs show a dark backdrop that makes the **Popup** stand out from other content on the page.

If you want to make a dialog truly modal, also set the **hideTrigger** property to **None**, so users won't be able to click the backdrop to dismiss the dialog. In this case, the dialog will close only if the **hide** method is called or if the user presses the Escape key.

Type
boolean

owner

Gets or sets the element that owns this **Popup**.

If the **owner** is null, the **Popup** behaves like a dialog. It is centered on the screen and must be shown using the **show** method.

Type
HTMLElement

removeOnHide

Gets or sets a value that determines whether the **Popup** element should be removed from the DOM when the **Popup** is hidden, as opposed to being hidden.

This property is set to true by default.

Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showTrigger

Gets or sets the actions that show the **`Popup`**.

By default, the **`showTrigger`** property is set to **`Click`**, which causes the popup to appear when the user clicks the owner element.

If you set the **`showTrigger`** property to **`None`**, the popup will be shown only when the **`show`** method is called.

Type
`PopupTrigger`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hide

```
hide(dialogResult?: any): void
```

Hides the **Popup**.

Parameters

- **dialogResult**: **any** OPTIONAL
  Optional value assigned to the **`dialogResult`** property before closing the **Popup**.

Returns
void

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onHidden

```
onHidden(e?: EventArgs): void
```

Raises the **hidden** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onHiding

```
onHiding(e: CancelEventArgs): boolean
```

Raises the **hiding** event.

Parameters

- **e**: **CancelEventArgs**

Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onShowing

```
onShowing(e: CancelEventArgs): boolean
```

Raises the **showing** event.

Parameters

- **e**: **CancelEventArgs**

Returns
boolean

onShown

```
onShown(e?: EventArgs): void
```

Raises the **shown** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

show

```
show(modal?: boolean, handleResult?: Function): void
```

Shows the **Popup**.

Parameters

- **modal**: **boolean** OPTIONAL
  Whether to show the popup as a modal dialog. If provided, this sets the value of the **modal** property.

- **handleResult**: **Function** OPTIONAL
  Callback invoked when the popup is hidden. If provided, this should be a function that receives the popup as a parameter.

  The **handleResult** callback allows callers to handle the result of modal dialogs without attaching handlers to the **hidden** event. For example, the code below shows a dialog used to edit the current item in a **CollectionView**. The edits are committed or canceled depending on the **dialogResult** value. For example:

```
$scope.editCurrentItem = function () {
  $scope.data.editItem($scope.data.currentItem);
  $scope.itemEditor.show(true, function (e) {
    if (e.dialogResult == 'wj-hide-ok') {
      $scope.data.commitEdit();
    } else {
      $scope.data.cancelEdit();
    }
  });
}
```

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

hidden

Occurs after the **Popup** has been hidden.

Arguments
`EventArgs`

hiding

Occurs before the **Popup** is hidden.

Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

showing

Occurs before the **Popup** is shown.

Arguments
`CancelEventArgs`

shown

Occurs after the **Popup** has been shown.

Arguments
`EventArgs`

DateSelectionMode Enum

File
`wijmo.input.js`
Module
wijmo.input

Specifies constants that define the date selection behavior.

Members

Name	Value	Description
None	0	The user cannot change the current value using the mouse or keyboard.
Day	1	The user can select days.
Month	2	The user can select months.

PopupTrigger Enum

File
`wijmo.input.js`
Module
wijmo.input

Specifies actions that trigger showing and hiding **Popup** controls.

Members

Name	Value	Description
None	0	No triggers; popups must be shown and hidden using code.
Click	1	Show or hide the popup when the owner element is clicked.
Blur	2	Hide the popup when it loses focus.
ClickOrBlur	Click | Blur	Show or hide the popup when the owner element is clicked, hide when it loses focus.

wijmo.chart Module

File
`wijmo.chart.js`
Module
wijmo.chart

Defines the **FlexChart** control and its associated classes.

The example below creates a **FlexChart** control and binds it to a data array. The chart has three series, each corresponding to a property in the objects contained in the source array.

The last series in the example uses the **chartType** property to override the default chart type used by the other series.

Example

Show me (http://jsfiddle.net/Wijmo5/6GB66)

Classes

- Axis
- AxisCollection
- ChartTooltip
- DataLabel
- DataLabelBase
- DataLabelRenderEventArgs
- DataPoint
- FlexChart
- FlexChartBase
- FlexChartCore
- FlexPie
- HitTestInfo
- Legend
- LineMarker
- Palettes
- PieDataLabel
- PlotArea
- PlotAreaCollection
- RenderEventArgs
- Series
- SeriesBase
- SeriesEventArgs
- SeriesRenderingEventArgs

Interfaces

- IRenderEngine

Enums

- AxisType
- ChartElement
- ChartType
- ImageFormat
- LabelPosition
- LineMarkerAlignment
- LineMarkerInteraction
- LineMarkerLines
- Marker
- OverlappingLabels
- PieLabelPosition
- Position
- SelectionMode
- SeriesVisibility
- Stacking
- TickMark

Axis Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
FlexRadarAxis, **WjFlexChartAxis**

Represents an axis in the chart.

Constructor

- constructor

Properties

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- reversed
- title

Methods

- convert
- convertBack
- onRangeChanged

Events

- rangeChanged

Constructor

constructor

```
constructor(position?: Position): Axis
```

Initializes a new instance of the **Axis** class.

Parameters

- **position**: **Position** OPTIONAL
  The position of the axis on the chart.

Returns
Axis

Properties

actualMax

Gets the actual axis maximum.

It returns a number or a Date object (for time-based data).

Type
any

actualMin

Gets the actual axis minimum.

It returns a number or a Date object (for time-based data).

Type
any

axisLine

Gets or sets a value indicating whether the axis line is visible.

Type
boolean

axisType

Gets the axis type.

Type
`AxisType`

binding

Gets or sets the comma-separated property names for the **`itemsSource`** property to use in axis labels.

The first name specifies the value on the axis, the second represents the corresponding axis label. The default value is 'value,text'.

Type
string

format

Gets or sets the format string used for the axis labels (see **`Globalize`**).

Type
string

hostElement

Gets the axis host element.

Type
SVGGElement

itemFormatter

Gets or sets the itemFormatter function for the axis labels.

If specified, the function takes two parameters:

- **render engine**: The **IRenderEngine** object to be used in formatting the labels.
- **current label**: An object with the following properties:
  - **value**: The value of the axis label to format.
  - **text**: The text to use in the label.
  - **pos**: The position in control coordinates at which the label is to be rendered.
  - **cls**: The CSS class to be applied to the label.

The function returns the label parameters of labels for which properties are modified.

For example:

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // red text
        label.cls = null; // no default CSS
     }
    return label;
}
```

Type
Function

itemsSource

Gets or sets the items source for the axis labels.

Names of the properties are specified by the **`binding`** property.

For example:

```
// default value for Axis.binding is 'value,text'
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

Type
any

labelAlign

Gets or sets the label alignment.

By default the labels are centered. The supported values are 'left' and 'right for x-axis and 'top' and 'bottom' for y-axis.

Type
string

labelAngle

Gets or sets the rotation angle of the axis labels.

The angle is measured in degrees with valid values ranging from -90 to 90.

Type
number

labelPadding

Gets or sets the label padding.

Type
number

labels

Gets or sets a value indicating whether the axis labels are visible.

Type
boolean

logBase

Gets or sets the logarithmic base of the axis.

If the base is not specified the axis uses a linear scale.

Use the **`logBase`** property to spread data that is clustered around the origin. This is common in several financial and economic data sets.

Type
number

majorGrid

Gets or sets a value indicating whether the axis includes grid lines.

Type
boolean

majorTickMarks

Gets or sets the location of the axis tick marks.

Type
`TickMark`

majorUnit

Gets or sets the number of units between axis labels.

If the axis contains date values, then the units are expressed in days.

Type
number

max

Gets or sets the maximum value shown on the axis.

If not set, the maximum is calculated automatically. The value can be a number or a Date object (for time-based data).

Type
any

min

Gets or sets the minimum value shown on the axis.

If not set, the minimum is calculated automatically. The value can be a number or a Date object (for time-based data).

Type
any

minorGrid

Gets or sets a value indicating whether the axis includes minor grid lines.

Type
boolean

minorTickMarks

Gets or sets the location of the minor axis tick marks.

Type
`TickMark`

minorUnit

Gets or sets the number of units between minor axis ticks.

If the axis contains date values, then the units are expressed in days.

Type
number

name

Gets or sets the axis name.

Type
string

origin

Gets or sets the value at which an axis crosses the perpendicular axis.

Type
number

overlappingLabels

Gets or sets a value indicating how to handle the overlapping axis labels.

Type
`OverlappingLabels`

plotArea

Gets or sets the plot area for the axis.

Type
`PlotArea`

position

Gets or sets the position of the axis with respect to the plot area.

Type
`Position`

reversed

Gets or sets a value indicating whether the axis is reversed (top to bottom or right to left).

Type
boolean

title

Gets or sets the title text shown next to the axis.

Type
string

Methods

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

Converts the specified value from data to pixel coordinates.

Parameters

- **val**: **number**
  The data value to convert.
- **maxValue**: **number** OPTIONAL
  The max value of the data, it's optional.
- **minValue**: **number** OPTIONAL
  The min value of the data, it's optional.

Returns
number

convertBack

```
convertBack(val: number): number
```

Converts the specified value from pixel to data coordinates.

Parameters

- **val**: **number**
  The pixel coordinates to convert back.

Returns
number

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

Raises the **rangeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

Events

rangeChanged

Occurs when the axis range changes.

Arguments
EventArgs

AxisCollection Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
ObservableArray

Represents a collection of **Axis** objects in a **FlexChart** control.

Constructor

- constructor

Properties

- isUpdating

Methods

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getAxis
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(data?: any[]): ObservableArray
```

Initializes a new instance of the **ObservableArray** class.

Parameters

- **data**: **any**[] OPTIONAL
  Array containing items used to populate the **ObservableArray**.

Inherited From
ObservableArray
Returns
ObservableArray

Properties

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **beginUpdate** and **endUpdate**).

Inherited From
ObservableArray
Type

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
ObservableArray
Returns
void

clear

```
clear(): void
```

Removes all items from the array.

Inherited From
ObservableArray
Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Inherited From
ObservableArray
Returns
void

getAxis

```
getAxis(name: string): Axis
```

Gets an axis by name.

Parameters

- **name**: **string**
  The name of the axis to look for.

Returns
Axis

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
ObservableArray
Returns
boolean

indexOf

```
indexOf(name: string): number
```

Gets the index of an axis by name.

Parameters

- **name**: **string**
  The name of the axis to look for.

Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Inherited From
ObservableArray
Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
ObservableArray
Returns
void

push

```
push(...item: any[]): number
```

Adds one or more items to the end of the array.

Parameters

- **...item**: **any**[]
  One or more items to add to the array.

Inherited From
ObservableArray
Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
ObservableArray
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Inherited From
ObservableArray
Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
ObservableArray
Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
ObservableArray
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes and/or adds items to the array.

Parameters

- **index**: **number**
  Position where items will be added or removed.
- **count**: **number**
  Number of items to remove from the array.
- **item**: **any** OPTIONAL
  Item to add to the array.

Inherited From
`ObservableArray`
Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

ChartTooltip Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
Tooltip

Extends the **Tooltip** class to provide chart tooltips.

Constructor

- constructor

Properties

- content
- gap
- hideDelay
- isContentHtml
- isVisible
- showAtMouse
- showDelay
- threshold

Methods

- dispose
- getTooltip
- hide
- onPopup
- setTooltip
- show

Events

- popup

Constructor

constructor

```
constructor(): ChartTooltip
```

Initializes a new instance of the **ChartTooltip** class.

Returns
ChartTooltip

Properties

content

Gets or sets the tooltip content.

The tooltip content can be specified as a string or as a function that takes a **HitTestInfo** object as a parameter.

When the tooltip content is a string, it may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

Parameters must be enclosed in single curly brackets. For example:

```
// 'country' and 'sales' are properties of the data object.
  chart.tooltip.content = '{country}, sales:{sales}';
```

The next example shows how to set the tooltip content using a function.

```
// Set the tooltip content
  chart.tooltip.content = function (ht) {
    return ht.name + ":" + ht.value.toFixed();
  }
```

Type
any

gap

Gets or sets the distance between the tooltip and the target element.

Inherited From
Tooltip
Type
number

hideDelay

Gets or sets the delay, in milliseconds, before hiding the tooltip after the mouse leaves the target element.

Inherited From
`Tooltip`
Type
number

isContentHtml

Gets or sets a value that determines whether the tooltip contents should be displayed as plain text or as HTML.

Inherited From
`Tooltip`
Type
boolean

isVisible

Gets a value that determines whether the tooltip is currently visible.

Inherited From
`Tooltip`
Type
boolean

showAtMouse

Gets or sets a value that determines whether the tooltip should be positioned with respect to the mouse position rather than the target element.

Inherited From
`Tooltip`
Type
boolean

showDelay

Gets or sets the delay, in milliseconds, before showing the tooltip after the mouse enters the target element.

Inherited From
`Tooltip`
Type
number

threshold

Gets or sets the maximum distance from the element to display the tooltip.

Type
number

Methods

dispose

```
dispose(): void
```

Removes all tooltips associated with this **`Tooltip`** instance.

Inherited From
`Tooltip`
Returns
void

getTooltip

```
getTooltip(element: any): string
```

Gets the tooltip content associated with a given element.

Parameters

- **element**: **any**
  Element, element ID, or control that the tooltip explains.

Inherited From
Tooltip
Returns
string

hide

```
hide(): void
```

Hides the tooltip if it is currently visible.

Inherited From
Tooltip
Returns
void

onPopup

```
onPopup(e: TooltipEventArgs): boolean
```

Raises the **popup** event.

Parameters

- **e**: **TooltipEventArgs**
  TooltipEventArgs that contains the event data.

Inherited From
Tooltip
Returns
boolean

setTooltip

```
setTooltip(element: any, content: string): void
```

Assigns tooltip content to a given element on the page.

The same tooltip may be used to display information for any number of elements on the page. To remove the tooltip from an element, call **setTooltip** and specify null for the content.

Parameters

- **element**: **any**
  Element, element ID, or control that the tooltip explains.

- **content**: **string**
  Tooltip content or ID of the element that contains the tooltip content.

Inherited From
Tooltip
Returns
void

show

```
show(element: any, content: string, bounds?: Rect): void
```

Shows the tooltip with the specified content, next to the specified element.

Parameters

- **element**: **any**
  Element, element ID, or control that the tooltip explains.

- **content**: **string**
  Tooltip content or ID of the element that contains the tooltip content.

- **bounds**: **Rect** OPTIONAL
  Optional element that defines the bounds of the area that the tooltip targets. If not provided, the bounds of the element are used (as reported by the **getBoundingClientRect** method).

Inherited From
Tooltip
Returns
void

Events

popup

Occurs before the tooltip content is displayed.

The event handler may customize the tooltip content or suppress the tooltip display by changing the event parameters.

Inherited From
`Tooltip`
Arguments
`TooltipEventArgs`

DataLabel Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
DataLabelBase
Derived Classes
WjFlexChartDataLabel

The point data label for FlexChart.

Properties

- border
- connectingLine
- content
- offset
- position

Methods

- onRendering

Events

- rendering

Properties

border

Gets or sets a value indicating whether the data labels have borders.

Inherited From
`DataLabelBase`
Type
boolean

connectingLine

Gets or sets a value indicating whether to draw lines that connect labels to the data points.

Inherited From
`DataLabelBase`
Type
boolean

content

Gets or sets the content of data labels.

The content can be specified as a string or as a function that takes **HitTestInfo** object as a parameter.

When the label content is a string, it can contain any of the following parameters:

- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point.
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point.
- **propertyName**: any property of data object.

The parameter must be enclosed in curly brackets, for example 'x={x}, y={y}'.

In the following example, we show the y value of the data point in the labels.

```
// Create a chart and show y data in labels positioned above the data point.
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

The next example shows how to set data label content using a function.

```
// Set the data label content
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

Inherited From
DataLabelBase
Type
any

offset

Gets or sets the offset from label to the data point.

Inherited From
`DataLabelBase`
Type
number

position

Gets or sets the position of the data labels.

Type
`LabelPosition`

Methods

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

Raises the **`rendering`** event.

Parameters

- **e**: **DataLabelRenderEventArgs**
  The **`DataLabelRenderEventArgs`** object used to render the label.

Inherited From
`DataLabelBase`
Returns
void

Events

rendering

Occurs before the data label is rendered.

Inherited From
`DataLabelBase`
Arguments
`DataLabelRenderEventArgs`

DataLabelBase Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
DataLabel, **PieDataLabel**

Represents the base abstract class for the **DataLabel** and the **PieDataLabel** classes.

Properties

- border
- connectingLine
- content
- offset

Methods

- onRendering

Events

- rendering

Properties

border

Gets or sets a value indicating whether the data labels have borders.

Type
boolean

connectingLine

Gets or sets a value indicating whether to draw lines that connect labels to the data points.

Type
boolean

content

Gets or sets the content of data labels.

The content can be specified as a string or as a function that takes **HitTestInfo** object as a parameter.

When the label content is a string, it can contain any of the following parameters:

- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point.
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point.
- **propertyName**: any property of data object.

The parameter must be enclosed in curly brackets, for example 'x={x}, y={y}'.

In the following example, we show the y value of the data point in the labels.

```
// Create a chart and show y data in labels positioned above the data point.
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

The next example shows how to set data label content using a function.

```
// Set the data label content
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

Type
any

offset

Gets or sets the offset from label to the data point.

Type
number

Methods

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

Raises the **`rendering`** event.

Parameters

- **e**: **DataLabelRenderEventArgs**
  The **`DataLabelRenderEventArgs`** object used to render the label.

Returns
void

Events

rendering

Occurs before the data label is rendered.

Arguments
`DataLabelRenderEventArgs`

DataLabelRenderEventArgs Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
RenderEventArgs

Provides arguments for **DataLabel** rendering event.

Constructor

- constructor

Properties

- cancel
- empty
- engine
- hitTestInfo
- point
- text

Constructor

constructor

```
constructor(engine: IRenderEngine, ht: HitTestInfo, pt: Point, text: string): DataLabelRenderEventArgs
```

Initializes a new instance of the **DataLabelRenderEventArgs** class.

Parameters

- **engine**: **IRenderEngine**
  (**IRenderEngine**) The rendering engine to use.
- **ht**: **HitTestInfo**
  The hit test information.
- **pt**: **Point**
  The reference point.
- **text**: **string**
  The label text.

Returns
DataLabelRenderEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be cancelled.

Type
boolean

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

engine

Gets the **`IRenderEngine`** object to use for rendering the chart elements.

Inherited From
`RenderEventArgs`
Type
`IRenderEngine`

hitTestInfo

Gets the hit test information.

Type
`HitTestInfo`

point

Gets the point associated with the label in control coordinates.

Type
`Point`

text

Gets or sets the label text.

Type
string

DataPoint Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
WjFlexChartDataPoint

Class that represents a data point (with x and y coordinates).

X and Y coordinates can be specified as a number or a Date object(time-based data).

Constructor

- constructor

Properties

- x
- y

Constructor

constructor

```
constructor(x?: any, y?: any): DataPoint
```

Initializes a new instance of the **`DataPoint`** class.

Parameters

- **x**: **any** OPTIONAL
  X coordinate of the new DataPoint.

- **y**: **any** OPTIONAL
  Y coordinate of the new DataPoint.

Returns
DataPoint

Properties

x

Gets or sets X coordinate value of this **`DataPoint`**.

Type
any

y

Gets or sets Y coordinate value of this **`DataPoint`**.

Type
any

FlexChart Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
FlexChartCore
Derived Classes
WjFlexChart

The **FlexChart** control provides a powerful and flexible way to visualize data.

You can use the **FlexChart** control to create charts that display data in several formats, including bar, line, symbol, bubble, and others.

To use the **FlexChart** control, set the **itemsSource** property to an array containing the data objects, then add one or more **Series** objects to the **series** property.

Use the **chartType** property to define the **ChartType** used as a default for all series. You may override the chart type for each series by setting the **chartType** property on the members of the **series** array.

Example

Show me (http://jsfiddle.net/Wijmo5/6GB66)

Constructor

- constructor

Properties

- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- options
- palette
- plotAreas
- plotMargin
- rightToLeft
- rotated
- selection
- selectionMode
- series
- stacking
- symbolSize
- tooltip

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

Constructor

constructor

```
constructor(element: any, options?): FlexChart
```

Initializes a new instance of the **FlexChart** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
FlexChart

Properties

axes

Gets the collection of **Axis** objects.

Inherited From
FlexChartCore
Type
ObservableArray

axisX

Gets or sets the main X axis.

Inherited From
FlexChartCore
Type
Axis

axisY

Gets or sets the main Y axis.

Inherited From
`FlexChartCore`
Type
`Axis`

binding

Gets or sets the name of the property that contains the Y values.

Inherited From
`FlexChartCore`
Type
string

bindingX

Gets or sets the name of the property that contains the X data values.

Inherited From
`FlexChartCore`
Type
string

chartType

Gets or sets the type of chart to create.

Type
`ChartType`

collectionView

Gets the **ICollectionView** object that contains the chart data.

Inherited From
FlexChartBase
Type
ICollectionView

dataLabel

Gets or sets the point data label.

Inherited From
FlexChartCore
Type
DataLabel

footer

Gets or sets the text displayed in the chart footer.

Inherited From
FlexChartBase
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
FlexChartBase
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

interpolateNulls

Gets or sets a value that determines whether to interpolate null values in the data.

If true, the chart interpolates the value of any missing data based on neighboring points. If false, it leaves a break in lines and areas at the points with null values.

Inherited From
`FlexChartCore`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

legendToggle

Gets or sets a value indicating whether clicking legend items toggles the series visibility in the chart.

Inherited From
`FlexChartCore`
Type
boolean

options

Gets or sets various chart options.

The following options are supported:

bubble.maxSize: Specifies the maximum size of symbols in the Bubble chart. The default value is 30 pixels.

bubble.minSize: Specifies the minimum size of symbols in the Bubble chart. The default value is 5 pixels.

```
chart.options = {
  bubble: { minSize: 5, maxSize: 30 }
}
```

funnel.neckWidth: Specifies the neck width as a percentage for the Funnel chart. The default value is 0.2.

funnel.neckHeight: Specifies the neck height as a percentage for the Funnel chart. The default value is 0.

funnel.type: Specifies the type of Funnel chart. It should be 'rectangle' or 'default'. neckWidth and neckHeight don't work if type is set to rectangle.

```
chart.options = {
  funnel: { neckWidth: 0.3, neckHeight: 0.3, type: 'rectangle' }
}
```

groupWidth: Specifies the group width for the Column charts, or the group height for the Bar charts. The group width can be specified in pixels or as percentage of the available space. The default value is '70%'.

```
chart.options = {
  groupWidth : 50; // 50 pixels
}
chart.options = {
  groupWidth : '100%'; // 100% pixels
}
```

Type
any

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **Palettes** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
FlexChartBase
Type
string[]

plotAreas

Gets the collection of **PlotArea** objects.

Inherited From
FlexChartCore
Type
PlotAreaCollection

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rotated

Gets or sets a value indicating whether to flip the axes so that X becomes vertical and Y becomes horizontal.

Type
boolean

selection

Gets or sets the selected chart series.

Inherited From
`FlexChartCore`
Type
`SeriesBase`

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

series

Gets the collection of **`Series`** objects.

Inherited From
`FlexChartCore`
Type
`ObservableArray`

stacking

Gets or sets a value that determines whether and how the series objects are stacked.

Type
`Stacking`

symbolSize

Gets or sets the size of the symbols used for all Series objects in this **FlexChart**.

This property may be overridden by the symbolSize property on each **Series** object.

Inherited From
FlexChartCore
Type
number

tooltip

Gets the chart **Tooltip** object.

The tooltip content is generated using a template that may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

To modify the template, assign a new value to the tooltip's content property. For example:

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

You can disable chart tooltips by setting the template to an empty string.

You can also use the **tooltip** property to customize tooltip parameters such as **showDelay** and **hideDelay**:

```
chart.tooltip.showDelay = 1000;
```

See **ChartTooltip** properties for more details and options.

Inherited From
FlexChartCore
Type
ChartTooltip

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Converts a **Point** from data coordinates to control coordinates.

Parameters

- **pt**: **any**
  Point in data coordinates, or X coordinate of a point in data coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Inherited From
FlexChartCore
Returns
Point

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

Raises the **seriesVisibilityChanged** event.

Parameters

- **e**: **SeriesEventArgs**
  The **SeriesEventArgs** object that contains the event data.

Inherited From
FlexChartCore
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
`FlexChartBase`
Returns
`Point`

pointToData

```
pointToData(pt: any, y?: number): Point
```

Converts a **`Point`** from control coordinates to chart data coordinates.

Parameters

- **pt**: **any**
  The point to convert, in control coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

seriesVisibilityChanged

Occurs when the series visibility changes, for example when the legendToggle property is set to true and the user clicks the legend.

Inherited From
`FlexChartCore`
Arguments
`SeriesEventArgs`

FlexChartBase Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
Control
Derived Classes
FlexChartCore, **FlexPie**, **TreeMap**

The **FlexChartBase** control from which the FlexChart and FlexPie derive.

Constructor

- constructor

Properties

- collectionView
- footer
- footerStyle
- header
- headerStyle
- hostElement
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- palette
- plotMargin
- rightToLeft
- selectionMode

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged

Constructor

constructor

```
constructor(element: any, options?, invalidateOnResize?: boolean): Control
```

Initializes a new instance of the **Control** class and attaches it to a DOM element.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.
- **invalidateOnResize**: **boolean** OPTIONAL
  Whether the control should be invalidated when it is resized.

Inherited From
Control
Returns
Control

Properties

collectionView

Gets the **ICollectionView** object that contains the chart data.

Type
ICollectionView

footer

Gets or sets the text displayed in the chart footer.

Type
string

footerStyle

Gets or sets the style of the chart footer.

Type
any

header

Gets or sets the text displayed in the chart header.

Type
string

headerStyle

Gets or sets the style of the chart header.

Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Type
any

legend

Gets or sets the chart legend.

Type
`Legend`

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **`Palettes`** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Type
string[]

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Type
any

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Type
`SelectionMode`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Returns
Point

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Arguments
`EventArgs`

FlexChartCore Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
FlexChartBase
Derived Classes
FlexChart, **FlexRadar**, **FinancialChart**

The core charting control for **FlexChart**.

Constructor

- constructor

Properties

- axes
- axisX
- axisY
- binding
- bindingX
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- palette
- plotAreas
- plotMargin
- rightToLeft
- selection
- selectionMode
- series
- symbolSize
- tooltip

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

Constructor

constructor

```
constructor(element: any, options?): FlexChartCore
```

Initializes a new instance of the **FlexChart** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
FlexChartCore

Properties

axes

Gets the collection of **Axis** objects.

Type
ObservableArray

axisX

Gets or sets the main X axis.

Type
Axis

axisY

Gets or sets the main Y axis.

Type
Axis

binding

Gets or sets the name of the property that contains the Y values.

Type
string

bindingX

Gets or sets the name of the property that contains the X data values.

Type
string

collectionView

Gets the **`ICollectionView`** object that contains the chart data.

Inherited From
`FlexChartBase`
Type
`ICollectionView`

dataLabel

Gets or sets the point data label.

Type
`DataLabel`

footer

Gets or sets the text displayed in the chart footer.

Inherited From
`FlexChartBase`
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
`FlexChartBase`
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

interpolateNulls

Gets or sets a value that determines whether to interpolate null values in the data.

If true, the chart interpolates the value of any missing data based on neighboring points. If false, it leaves a break in lines and areas at the points with null values.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

legendToggle

Gets or sets a value indicating whether clicking legend items toggles the series visibility in the chart.

Type
boolean

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **`Palettes`** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
`FlexChartBase`
Type
string[]

plotAreas

Gets the collection of **`PlotArea`** objects.

Type
`PlotAreaCollection`

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selection

Gets or sets the selected chart series.

Type
`SeriesBase`

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

series

Gets the collection of **`Series`** objects.

Type
`ObservableArray`

symbolSize

Gets or sets the size of the symbols used for all Series objects in this **`FlexChart`**.

This property may be overridden by the symbolSize property on each **`Series`** object.

Type
number

tooltip

Gets the chart **Tooltip** object.

The tooltip content is generated using a template that may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

To modify the template, assign a new value to the tooltip's content property. For example:

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

You can disable chart tooltips by setting the template to an empty string.

You can also use the **tooltip** property to customize tooltip parameters such as **showDelay** and **hideDelay**:

```
chart.tooltip.showDelay = 1000;
```

See **ChartTooltip** properties for more details and options.

Type
ChartTooltip

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Converts a **Point** from data coordinates to control coordinates.

Parameters

- **pt**: **any**
  Point in data coordinates, or X coordinate of a point in data coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Returns
Point

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

Raises the **seriesVisibilityChanged** event.

Parameters

- **e**: **SeriesEventArgs**
  The **SeriesEventArgs** object that contains the event data.

Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
`FlexChartBase`
Returns
`Point`

pointToData

```
pointToData(pt: any, y?: number): Point
```

Converts a **`Point`** from control coordinates to chart data coordinates.

Parameters

- **pt**: **any**
  The point to convert, in control coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Returns
`Point`

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

seriesVisibilityChanged

Occurs when the series visibility changes, for example when the legendToggle property is set to true and the user clicks the legend.

Arguments
`SeriesEventArgs`

FlexPie Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
FlexChartBase
Derived Classes
Sunburst, **WjFlexPie**

The **FlexPie** control provides pie and doughnut charts with selectable slices.

To use the **FlexPie** control, set the **itemsSource** property to an array containing the data and use the **binding** and **bindingName** properties to set the properties that contain the item values and names.

Constructor

- constructor

Properties

- binding
- bindingName
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- innerRadius
- isAnimated
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- offset
- palette
- plotMargin
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition
- selectionMode
- startAngle
- tooltip

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged

Constructor

constructor

```
constructor(element: any, options?): FlexPie
```

Initializes a new instance of the **FlexPie** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A Javascript object containing initialization data for the control.

Returns
FlexPie

Properties

binding

Gets or sets the name of the property that contains the chart values.

Type
string

bindingName

Gets or sets the name of the property that contains the name of the data items.

Type
string

collectionView

Gets the **ICollectionView** object that contains the chart data.

Inherited From
FlexChartBase
Type
ICollectionView

dataLabel

Gets or sets the point data label.

Type
`PieDataLabel`

footer

Gets or sets the text displayed in the chart footer.

Inherited From
`FlexChartBase`
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
`FlexChartBase`
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

innerRadius

Gets or sets the size of the pie's inner radius.

The inner radius is measured as a fraction of the pie radius.

The default value for this property is zero, which creates a pie. Setting this property to values greater than zero creates pies with a hole in the middle, also known as doughnut charts.

Type
number

isAnimated

Gets or sets a value indicating whether to use animation when items are selected.

See also the **`selectedItemPosition`** and **`selectionMode`** properties.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

offset

Gets or sets the offset of the slices from the pie center.

The offset is measured as a fraction of the pie radius.

Type
number

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **`Palettes`** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
`FlexChartBase`
Type
string[]

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

reversed

Gets or sets a value that determines whether angles are reversed (counter-clockwise).

The default value is false, which causes angles to be measured in the clockwise direction.

Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the selected slice.

Type
number

selectedItemOffset

Gets or sets the offset of the selected slice from the pie center.

Offsets are measured as a fraction of the pie radius.

Type
number

selectedItemPosition

Gets or sets the position of the selected slice.

Setting this property to a value other than 'None' causes the pie to rotate when an item is selected.

Note that in order to select slices by clicking the chart, you must set the **`selectionMode`** property to "Point".

Type
`Position`

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

startAngle

Gets or sets the starting angle for the pie slices, in degrees.

Angles are measured clockwise, starting at the 9 o'clock position.

Type
number

tooltip

Gets the chart's **`Tooltip`**.

Type
`ChartTooltip`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
FlexChartBase
Returns
Point

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

HitTestInfo Class

File
`wijmo.chart.js`
Module
wijmo.chart

Contains information about a part of a **FlexChart** control at a specified page coordinate.

Constructor

- constructor

Properties

- chartElement
- distance
- item
- point
- pointIndex
- series
- x
- y

Constructor

constructor

```
constructor(chart: FlexChartBase, point: Point, element?: ChartElement): HitTestInfo
```

Initializes a new instance of the **HitTestInfo** class.

Parameters

- **chart**: **FlexChartBase**
  The chart control.
- **point**: **Point**
  The original point in window coordinates.
- **element**: **ChartElement** OPTIONAL
  The chart element.

Returns
HitTestInfo

Properties

chartElement

Gets the chart element at the specified coordinates.

Type
`ChartElement`

distance

Gets the distance from the closest data point.

Type
number

item

Gets the data object that corresponds to the closest data point.

Type
any

point

Gets the point in control coordinates to which this `HitTestInfo` object refers to.

Type
`Point`

pointIndex

Gets the data point index at the specified coordinates.

Type
number

series

Gets the chart series at the specified coordinates.

Type
`SeriesBase`

x

Gets the x-value of the closest data point.

Type
any

y

Gets the y-value of the closest data point.

Type
any

Legend Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
WjFlexChartLegend

Represents the chart legend.

Constructor

- constructor

Properties

- position

Constructor

constructor

```
constructor(chart: FlexChartBase): Legend
```

Initializes a new instance of the **Legend** class.

Parameters

- **chart**: **FlexChartBase**
  `FlexChartBase` that owns this **Legend**.

Returns
Legend

Properties

position

Gets or sets a value that determines whether and where the legend appears in relation to the plot area.

Type
`Position`

LineMarker Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
WjFlexChartLineMarker

Represents an extension of the LineMarker for the FlexChart.

The **LineMarker** consists of a text area with content reflecting data point values, and optional vertical or horizontal lines (or both for a cross-hair effect) positioned over the plot area.

It can be static (interaction = None), follow the mouse or touch position (interaction = Move), or move when the user drags the line (interaction = Drag).

For example:

```
// create an interactive marker with a horizontal line and y-value
  var lm = new wijmo.chart.LineMarker($scope.ctx.chart, {
      lines: wijmo.chart.LineMarkerLines.Horizontal,
      interaction: wijmo.chart.LineMarkerInteraction.Move,
      alignment : wijmo.chart.LineMarkerAlignment.Top
  });
  lm.content = function (ht) {

      // show y-value
      return lm.y.toFixed(2);
  }
```

Constructor

- constructor

Properties

- alignment
- chart
- content
- dragContent
- dragLines
- dragThreshold
- horizontalPosition
- interaction
- isVisible
- lines
- seriesIndex
- verticalPosition
- x
- y

Methods

- onPositionChanged
- remove

Events

- positionChanged

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): LineMarker
```

Initializes a new instance of the **LineMarker** class.

Parameters

- **chart**: **FlexChartCore**
  The chart on which the LineMarker appears.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
LineMarker

Properties

alignment

Gets or sets the alignment of the LineMarker content.

By default, the LineMarker shows to the right, at the bottom of the target point. Use '|' to combine alignment values.

```
// set the alignment to the left.
marker.alignment = wijmo.chart.LineMarkerAlignment.Left;

// set the alignment to the left top.
marker.alignment = wijmo.chart.LineMarkerAlignment.Left | wijmo.chart.LineMarkerAlignment.Top;
```

Type
`LineMarkerAlignment`

chart

Gets the **`FlexChart`** object that owns the LineMarker.

Type
`FlexChartCore`

content

Gets or sets the content function that allows you to customize the text content of the LineMarker.

Type
Function

dragContent

Gets or sets a value indicating whether the content of the marker is draggable when the interaction mode is "Drag."

Type
boolean

dragLines

Gets or sets a value indicating whether the lines are linked when the horizontal or vertical line is dragged when the interaction mode is "Drag."

Type
boolean

dragThreshold

Gets or sets the maximum distance from the horizontal or vertical line that the marker can be dragged.

Type
number

horizontalPosition

Gets or sets the horizontal position of the LineMarker relative to the plot area.

Its value range is (0, 1). If the value is null or undefined and **`interaction`** is set to wijmo.chart.LineMarkerInteraction.Move or wijmo.chart.LineMarkerInteraction.Drag, the horizontal position of the marker is calculated automatically based on the pointer's position.

Type
number

interaction

Gets or sets the interaction mode of the LineMarker.

Type
`LineMarkerInteraction`

isVisible

Gets or sets the visibility of the LineMarker.

Type
boolean

lines

Gets or sets the visibility of the LineMarker lines.

Type
`LineMarkerLines`

seriesIndex

Gets or sets the index of the series in the chart in which the LineMarker appears. This takes effect when the **`interaction`** property is set to wijmo.chart.LineMarkerInteraction.Move or wijmo.chart.LineMarkerInteraction.Drag.

Type
number

verticalPosition

Gets or sets the vertical position of the LineMarker relative to the plot area.

Its value range is (0, 1). If the value is null or undefined and **`interaction`** is set to wijmo.chart.LineMarkerInteraction.Move or wijmo.chart.LineMarkerInteraction.Drag, the vertical position of the LineMarker is calculated automatically based on the pointer's position.

Type
number

x

Gets the current x-value as chart data coordinates.

Type
number

y

Gets the current y-value as chart data coordinates.

Type
number

Methods

onPositionChanged

```
onPositionChanged(point: Point): void
```

Raises the **positionChanged** event.

Parameters

- **point**: **Point**
  The target point at which to show the LineMarker.

Returns
void

remove

```
remove(): void
```

Removes the LineMarker from the chart.

Returns
void

Events

positionChanged

Occurs after the **LineMarker**'s position changes.

Arguments
Point

Palettes Class

File
`wijmo.chart.js`
Module
wijmo.chart

These are predefined color palettes for chart **Series** objects.

To create custom color palettes, supply an array of strings or rgba values.

You can specify palettes for **FlexChart** and **FlexPie** controls. For example:

```
chart.palette = Palettes.light;
```

The following palettes are pre-defined:

- standard (default)
- cocoa
- coral
- dark
- highcontrast
- light
- midnight
- modern
- organic
- slate
- zen
- cyborg
- superhero
- flatly
- darkly
- cerulan

PieDataLabel Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
DataLabelBase
Derived Classes
WjFlexPieDataLabel

The point data label for FlexPie.

Properties

- border
- connectingLine
- content
- offset
- position

Methods

- onRendering

Events

- rendering

Properties

border

Gets or sets a value indicating whether the data labels have borders.

Inherited From
`DataLabelBase`
Type
boolean

connectingLine

Gets or sets a value indicating whether to draw lines that connect labels to the data points.

Inherited From
`DataLabelBase`
Type
boolean

content

Gets or sets the content of data labels.

The content can be specified as a string or as a function that takes **HitTestInfo** object as a parameter.

When the label content is a string, it can contain any of the following parameters:

- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point.
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point.
- **propertyName**: any property of data object.

The parameter must be enclosed in curly brackets, for example 'x={x}, y={y}'.

In the following example, we show the y value of the data point in the labels.

```
// Create a chart and show y data in labels positioned above the data point.
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

The next example shows how to set data label content using a function.

```
// Set the data label content
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

Inherited From
DataLabelBase
Type
any

offset

Gets or sets the offset from label to the data point.

Inherited From
`DataLabelBase`
Type
number

position

Gets or sets the position of the data labels.

Type
`PieLabelPosition`

Methods

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

Raises the **`rendering`** event.

Parameters

- **e**: **DataLabelRenderEventArgs**
  The **`DataLabelRenderEventArgs`** object used to render the label.

Inherited From
`DataLabelBase`
Returns
void

Events

rendering

Occurs before the data label is rendered.

Inherited From
`DataLabelBase`
Arguments
`DataLabelRenderEventArgs`

PlotArea Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
WjFlexChartPlotArea

Represents a plot area on the chart.

The chart can have multiple plot areas with multiple axes. To assign axis to plot area use **Axis.plotArea** property. For example:

```
// create a plot area
 var pa = new wijmo.chart.PlotArea();
 pa.row = 1;
 chart.plotAreas.push(pa);

 // create auxiliary y-axis
 var ay2 = new wijmo.chart.Axis(wijmo.chart.Position.Left);
 ay2.plotArea = pa; // attach axis to the plot area
 chart.axes.push(ay2);

 // plot first series along y-axis
 chart.series[0].axisY = ay2;
```

Constructor

- constructor

Properties

- column
- height
- name
- row
- style
- width

Constructor

constructor

```
constructor(options?: any): PlotArea
```

Initializes a new instance of the **PlotArea** class.

Parameters

- **options**: **any** OPTIONAL
  Initialization options for the plot area.

Returns
PlotArea

Properties

column

Gets or sets the column index of plot area. This determines the horizontal position of the plot area on the chart.

Type
number

height

Gets or sets the height of the plot area.

The height can be specified as a number (in pixels) or as a string in the format '{number}*' (star sizing).

Type
any

name

Gets or sets the plot area name.

Type
string

row

Gets or sets the row index of plot area. This determines the vertical position of the plot area on the chart.

Type
number

style

Gets or sets the style of the plot area.

Using **style** property, you can set appearance of the plot area. For example:

```
pa.style = { fill: 'rgba(0,255,0,0.1)' };
```

Type
any

width

Gets or sets width of the plot area.

The width can be specified as a number (in pixels) or as a string in the format '{number}*' (star sizing).

Type
any

PlotAreaCollection Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
ObservableArray

Represents a collection of **PlotArea** objects in a **FlexChartCore** control.

Constructor

- constructor

Properties

- isUpdating

Methods

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getPlotArea
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(data?: any[]): ObservableArray
```

Initializes a new instance of the **ObservableArray** class.

Parameters

- **data**: **any**[] OPTIONAL
  Array containing items used to populate the **ObservableArray**.

Inherited From
ObservableArray
Returns
ObservableArray

Properties

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **beginUpdate** and **endUpdate**).

Inherited From
ObservableArray
Type

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
ObservableArray
Returns
void

clear

```
clear(): void
```

Removes all items from the array.

Inherited From
ObservableArray
Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Inherited From
ObservableArray
Returns
void

getPlotArea

```
getPlotArea(name: string): PlotArea
```

Gets a plot area by name.

Parameters

- **name**: **string**
  The name of the plot area to look for.

Returns
PlotArea

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
ObservableArray
Returns
boolean

indexOf

```
indexOf(name: string): number
```

Gets the index of a plot area by name.

Parameters

- **name**: **string**
  The name of the plot area to look for.

Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Inherited From
ObservableArray
Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
ObservableArray
Returns
void

push

```
push(...item: any[]): number
```

Adds one or more items to the end of the array.

Parameters

- **...item**: **any**[]
  One or more items to add to the array.

Inherited From
ObservableArray
Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
ObservableArray
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Inherited From
ObservableArray
Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
ObservableArray
Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
ObservableArray
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes and/or adds items to the array.

Parameters

- **index**: **number**
  Position where items will be added or removed.
- **count**: **number**
  Number of items to remove from the array.
- **item**: **any** OPTIONAL
  Item to add to the array.

Inherited From
`ObservableArray`
Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

RenderEventArgs Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
CancelEventArgs
Derived Classes
DataLabelRenderEventArgs, **SeriesRenderingEventArgs**

Provides arguments for **Series** events.

Constructor

- constructor

Properties

- cancel
- empty
- engine

Constructor

constructor

```
constructor(engine: IRenderEngine): RenderEventArgs
```

Initializes a new instance of the **RenderEventArgs** class.

Parameters
- **engine**: **IRenderEngine**
  (**IRenderEngine**) The rendering engine to use.

Returns
RenderEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

engine

Gets the **`IRenderEngine`** object to use for rendering the chart elements.

Type
`IRenderEngine`

Series Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
SeriesBase
Derived Classes
ErrorBar, **WjFlexChartSeries**

Represents a series of data points to display in the chart.

The **Series** class supports all basic chart types. You may define a different chart type on each **Series** object that you add to the **FlexChart** series collection. This overrides the **chartType** property set on the chart that is the default for all **Series** objects in its collection.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SeriesBase
```

Initializes a new instance of the **SeriesBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
SeriesBase
Returns
SeriesBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart.

Type
`ChartType`

collectionView

Gets the `ICollectionView` object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

SeriesBase Class

File
`wijmo.chart.js`
Module
wijmo.chart
Derived Classes
Series, **BoxWhisker**, **TrendLineBase**, **Waterfall**, **FlexRadarSeries**, **FinancialSeries**, **Fibonacci**, **FibonacciArcs**, **FibonacciFans**, **FibonacciTimeZones**, **OverlayIndicatorBase**

Represents a series of data points to display in the chart.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SeriesBase
```

Initializes a new instance of the **SeriesBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
SeriesBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Type
any

axisX

Gets or sets the x-axis for the series.

Type
Axis

axisY

Gets or sets the y-axis for the series.

Type
Axis

binding

Gets or sets the name of the property that contains Y values for the series.

Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Type
string

chart

Gets the **`FlexChart`** object that owns this series.

Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Type
string

hostElement

Gets the series host element.

Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Type
any

legendElement

Gets the series element in the legend.

Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Type
string

style

Gets or sets the series style.

Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Returns
Point

Events

rendered

Occurs when series is rendered.

Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Arguments
`EventArgs`

SeriesEventArgs Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
EventArgs

Provides arguments for **Series** events.

Constructor

- constructor

Properties

- empty
- series

Constructor

constructor

```
constructor(series: SeriesBase): SeriesEventArgs
```

Initializes a new instance of the **SeriesEventArgs** class.

Parameters

- **series**: **SeriesBase**
  Specifies the **Series** object affected by this event.

Returns
SeriesEventArgs

Properties

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

series

Gets the **`Series`** object affected by this event.

Type
`SeriesBase`

SeriesRenderingEventArgs Class

File
`wijmo.chart.js`
Module
wijmo.chart
Base Class
RenderEventArgs

Provides arguments for **Series** rendering event.

Constructor

- constructor

Properties

- cancel
- count
- empty
- engine
- index

Constructor

constructor

```
constructor(engine: IRenderEngine, index: number, count: number): SeriesRenderingEventArgs
```

Initializes a new instance of the **SeriesRenderingEventArgs** class.

Parameters

- **engine**: **IRenderEngine**
  (**IRenderEngine**) The rendering engine to use.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Returns
SeriesRenderingEventArgs

Properties

-

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

count

Gets total number of series to render.

Type
number

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

engine

Gets the **`IRenderEngine`** object to use for rendering the chart elements.

Inherited From
`RenderEventArgs`
Type
`IRenderEngine`

index

Gets the index of the series to render.

Type
number

IRenderEngine Interface

File
`wijmo.chart.js`
Module
wijmo.chart

Represents a rendering engine that performs the basic drawing routines.

Properties

- element
- fill
- fontFamily
- fontSize
- stroke
- strokeWidth
- textFill

Methods

- addClipRect
- beginRender
- drawDonutSegment
- drawEllipse
- drawImage
- drawLine
- drawLines
- drawPieSegment
- drawPolygon
- drawRect
- drawSplines
- drawString
- drawStringRotated
- endGroup
- endRender
- measureString
- setViewportSize
- startGroup

Properties

element

Gets the rendered element.

Type
Element

fill

Gets or sets the color used to fill the element.

Type
string

fontFamily

Gets or sets the font family for the text output.

Type
string

fontSize

Gets or sets the font size for the text output.

Type
string

stroke

Gets or sets the color used to outline the element.

Type
string

strokeWidth

Gets or sets the thickness of the outline.

Type
number

textFill

Gets or sets the text color.

Type
string

Methods

addClipRect

```
addClipRect(clipRect: Rect, id: string): void
```

Adds a clipping rectangle to the context.

Parameters

- **clipRect**: **Rect**
  The clipping rectangle.

- **id**: **string**
  The ID of the clipping rectangle.

Returns
void

beginRender

```
beginRender(): void
```

Clears the viewport and starts the rendering cycle.

Returns
void

drawDonutSegment

```
drawDonutSegment(cx: number, cy: number, radius: number, innerRadius: number, startAngle: number, sweepAngle: number, className?: string, style?: any, clipPath?: string): void
```

Draws a doughnut segment.

Parameters

- **cx**: **number**
  X coordinate of the segment center.
- **cy**: **number**
  Y coordinate of the segment center.
- **radius**: **number**
  Outer radius of the segment.
- **innerRadius**: **number**
  Inner radius of the segment.
- **startAngle**: **number**
  Start angle of the segment, in degrees.
- **sweepAngle**: **number**
  Sweep angle of the segment, in degrees clockwise.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.
- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

Returns
void

drawEllipse

```
drawEllipse(cx: number, cy: number, rx: number, ry: number, className?: string, style?: any): void
```

Draws an ellipse.

Parameters

- **cx**: **number**
  X coordinate of the ellipse's center.
- **cy**: **number**
  Y coordinate of the ellipse's center.
- **rx**: **number**
  X radius (half of the ellipse's width).
- **ry**: **number**
  Y radius (half of the ellipse's height).
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.

Returns
void

drawImage

```
drawImage(href: string, x: number, y: number, w: number, h: number): void
```

Draws an image.

Parameters

- **href**: **string**
  Url of the image to draw.
- **x**: **number**
  Left coordinate of the image's bounding rectangle.
- **y**: **number**
  Bottom coordinate of the image's bounding rectangle.
- **w**: **number**
  Image width.
- **h**: **number**
  Image height.

Returns
void

drawLine

```
drawLine(x1: number, y1: number, x2: number, y2: number, className?: string, style?: any): void
```

Draws a line.

Parameters

- **x1**: **number**
  X coordinate of the first point.
- **y1**: **number**
  Y coordinate of the first point.
- **x2**: **number**
  X coordinate of the second point.
- **y2**: **number**
  Y coordinate of the second point.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.

Returns
void

drawLines

```
drawLines(xs: number[], ys: number[], className?: string, style?: any, clipPath?: string): void
```

Draws a series of lines.

Parameters

- **xs**: **number**[]
  Array of X coordinates.
- **ys**: **number**[]
  Array of Y coordinates.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.
- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

Returns
void

drawPieSegment

```
drawPieSegment(cx: number, cy: number, radius: number, startAngle: number, sweepAngle: number, className?: string, style?: any, clipPath?: string): void
```

Draws a pie segment.

Parameters

- **cx**: **number**
  X coordinate of the segment center.
- **cy**: **number**
  Y coordinate of the segment center.
- **radius**: **number**
  Radius of the segment.
- **startAngle**: **number**
  Start angle of the segment, in degrees.
- **sweepAngle**: **number**
  Sweep angle of the segment, in degrees clockwise.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.
- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

Returns
void

drawPolygon

```
drawPolygon(xs: number[], ys: number[], className?: string, style?: any, clipPath?: string): void
```

Draws a polygon.

Parameters

- **xs**: **number**[]
  Array of X coordinates.
- **ys**: **number**[]
  Array of Y coordinates.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.
- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

Returns
void

drawRect

```
drawRect(x: number, y: number, w: number, h: number, className?: string, style?: any, clipPath?: string): void
```

Draws a rectangle.

Parameters

- **x**: **number**
  Left of the rectangle.
- **y**: **number**
  Bottom of the rectangle.
- **w**: **number**
  Width of the rectangle.
- **h**: **number**
  Height of the rectangle.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.
- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

Returns
void

drawSplines

```
drawSplines(xs: number[], ys: number[], className?: string, style?: any, clipPath?: string): void
```

Draws a series of splines (smooth path).

Parameters

- **xs**: **number**[]
  Array of X coordinates.
- **ys**: **number**[]
  Array of Y coordinates.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.
- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

Returns
void

drawString

```
drawString(s: string, pt: Point, className?: string, style?: any): void
```

Draws a string.

Parameters

- **s**: **string**
  String to be drawn.
- **pt**: **Point**
  Reference point for the string.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.

Returns
void

drawStringRotated

```
drawStringRotated(s: string, pt: Point, center: Point, angle: number, className?: string, style?: any): void
```

Draws a rotated string.

Parameters

- **s**: **string**
  String to be drawn.
- **pt**: **Point**
  Reference point for rendering the string.
- **center**: **Point**
  Reference point for rotating the string.
- **angle**: **number**
  Rotation angle, in degrees, clockwise.
- **className**: **string** OPTIONAL
  Class name to be applied to the element.
- **style**: **any** OPTIONAL
  Style object to be applied to the element.

Returns
void

endGroup

```
endGroup(): void
```

Ends a group.

Returns
void

endRender

```
endRender(): void
```

Finishes the rendering cycle.

Returns
void

measureString

```
measureString(s: string, className?: string, groupName?: string, style?: any): Size
```

Measures a string.

Parameters

- **s**: **string**
  String to be measured.
- **className**: **string** OPTIONAL
  Class name to use when measuring the string.
- **groupName**: **string** OPTIONAL
  Name of the group to use when measuring the string.
- **style**: **any** OPTIONAL
  Style object to use when measuring the string.

Returns
Size

setViewportSize

```
setViewportSize(w: number, h: number): void
```

Sets the size of the viewport.

Parameters

- **w**: **number**
  Viewport width.

- **h**: **number**
  Viewport height.

Returns
void

startGroup

```
startGroup(className?: string, clipPath?: string, createTransform?: boolean): void
```

Starts a group.

Parameters

- **className**: **string** OPTIONAL
  Class name to apply to the new group.

- **clipPath**: **string** OPTIONAL
  Id of the path to use as a clipping path.

- **createTransform**: **boolean** OPTIONAL
  Whether to create a new transform for the group.

Returns
void

AxisType Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the axis type.

Members

Name	Value	Description
X	0	Category axis (normally horizontal).
Y	1	Value axis (normally vertical).

ChartElement Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the type of chart element found by the hitTest method.

Members

Name	Value	Description
PlotArea	0	The area within the axes.
AxisX	1	X-axis.
AxisY	2	Y-axis.
ChartArea	3	The area within the control but outside the axes.
Legend	4	The chart legend.
Header	5	The chart header.
Footer	6	The chart footer.
Series	7	A chart series.
SeriesSymbol	8	A chart series symbol.
DataLabel	9	A data label.
None	10	No chart element.

ChartType Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the chart type.

Members

Name	Value	Description
Column	0	Shows vertical bars and allows you to compare values of items across categories.
Bar	1	Shows horizontal bars.
Scatter	2	Shows patterns within the data using X and Y coordinates.
Line	3	Shows trends over a period of time or across categories.
LineSymbols	4	Shows a line chart with a symbol on each data point.
Area	5	Shows a line chart with the area below the line filled with color.
Bubble	6	Shows a Scatter chart with a third data value that determines the size of the symbol. The data for this chart type can be defined using the **`FlexChart`** or **`Series`** **binding** property as a comma separated value in the following format: "yProperty, bubbleSizeProperty".
Candlestick	7	Presents items with high, low, open, and close values. The size of the wick line is determined by the High and Low values, while the size of the bar is determined by the Open and Close values. The bar is displayed using different colors, depending on whether the close value is higher or lower than the open value. The data for this chart type can be defined using the **`FlexChart`** or **`Series`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
HighLowOpenClose	8	Displays the same information as a candlestick chart, except that opening values are displayed using lines to the left, while lines to the right indicate closing values. The data for this chart type can be defined using the **`FlexChart`** or **`Series`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
Spline	9	Displays a line chart that plots curves rather than angled lines through the data points.
SplineSymbols	10	Displays a spline chart with symbols on each data point.
SplineArea	11	Displays a spline chart with the area below the line filled with color.
Funnel	12	Displays a funnel chart, usually representing stages in a process such as a sales pipeline.

ImageFormat Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the format of the image with embed base64-encoded binary data.

Members

Name	Value	Description
Png	0	Gets the W3C Portable Network Graphics (PNG) image format.
Jpeg	1	Gets the Joint Photographic Experts Group (JPEG) image format.
Svg	2	Gets the Scalable Vector Graphics(SVG) image format.

LabelPosition Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the position of data labels on the chart.

Members

Name	Value	Description
None	0	No data labels appear.
Left	1	The labels appear to the left of the data points.
Top	2	The labels appear above the data points.
Right	3	The labels appear to the right of the data points.
Bottom	4	The labels appear below the data points.
Center	5	The labels appear centered on the data points.

LineMarkerAlignment Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the alignment of the **LineMarker**.

Members

Name	Value	Description
Auto	2	The LineMarker alignment adjusts automatically so that it stays within the boundaries of the plot area.
Right	0	The LineMarker aligns to the right of the pointer.
Left	1	The LineMarker aligns to the left of the pointer.
Bottom	4	The LineMarker aligns to the bottom of the pointer.
Top	6	The LineMarker aligns to the top of the pointer.

LineMarkerInteraction Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies how the **LineMarker** interacts with the user.

Members

Name	Value	Description
None	0	No interaction, the user specifies the position by clicking.
Move	1	The **`LineMarker`** moves with the pointer.
Drag	2	The **`LineMarker`** moves when the user drags the lines.

LineMarkerLines Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the direction of the lines shown by the **LineMarker**.

Members

Name	Value	Description
None	0	No lines.
Vertical	1	Vertical line.
Horizontal	2	Horizontal line.
Both	3	Vertical and horizontal lines.

Marker Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the type of marker to use for the **symbolMarker** property.

Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Members

Name	Value	Description
Dot	0	Uses a circle to mark each data point.
Box	1	Uses a square to mark each data point.

OverlappingLabels Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies how to handle overlapping labels.

Members

Name	Value	Description
Auto	0	Hide overlapping labels.
Show	1	Show all labels, including overlapping ones.

PieLabelPosition Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the position of data labels on the pie chart.

Members

Name	Value	Description
None	0	No data labels.
Inside	1	The label appears inside the pie slice.
Center	2	The item appears at the center of the pie slice.
Outside	3	The item appears outside the pie slice.

Position Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies the position of an axis or legend on the chart.

Members

Name	Value	Description
None	0	The item is not visible.
Left	1	The item appears to the left of the chart.
Top	2	The item appears above the chart.
Right	3	The item appears to the right of the chart.
Bottom	4	The item appears below the chart.
Auto	5	The item is positioned automatically.

SelectionMode Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies what is selected when the user clicks the chart.

Members

Name	Value	Description
None	0	Select neither series nor data points when the user clicks the chart.
Series	1	Select the whole **`Series`** when the user clicks it on the chart.
Point	2	Select the data point when the user clicks it on the chart. Since Line, Area, Spline, and SplineArea charts do not render individual data points, nothing is selected with this setting on those chart types.

SeriesVisibility Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies whether and where the Series is visible.

Members

Name	Value	Description
Visible	0	The series is visible on the plot and in the legend.
Plot	1	The series is visible only on the plot.
Legend	2	The series is visible only in the legend.
Hidden	3	The series is hidden.

Stacking Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies whether and how to stack the chart's data values.

Members

Name	Value	Description
None	0	No stacking. Each series object is plotted independently.
Stacked	1	Stacked charts show how each value contributes to the total.
Stacked100pc	2	100% stacked charts show how each value contributes to the total with the relative size of each series representing its contribution to the total.

TickMark Enum

File
`wijmo.chart.js`
Module
wijmo.chart

Specifies whether and where the axis tick marks appear.

Members

Name	Value	Description
None	0	No tick marks appear.
Outside	1	Tick marks appear outside the plot area.
Inside	2	Tick marks appear inside the plot area.
Cross	3	Tick marks cross the axis.

wijmo.chart.analytics Module

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Defines classes that add analytics features to charts including **TrendLine**, **MovingAverage** and **FunctionSeries**.

Classes

- BoxWhisker
- ErrorBar
- FunctionSeries
- MovingAverage
- ParametricFunctionSeries
- TrendLine
- TrendLineBase
- Waterfall
- YFunctionSeries

Enums

- ErrorAmount
- ErrorBarDirection
- ErrorBarEndStyle
- MovingAverageType
- QuartileCalculation
- TrendLineFitType

BoxWhisker Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
SeriesBase
Derived Classes
WjFlexChartBoxWhisker

Represents a Box&Whisker chart series.

The **BoxWhisker** series is normally used to compare distributions between different sets of numerical data.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- gapWidth
- groupWidth
- hostElement
- itemsSource
- legendElement
- meanLineStyle
- meanMarkerStyle
- name
- quartileCalculation
- showInnerPoints
- showMeanLine
- showMeanMarker
- showOutliers
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): BoxWhisker
```

Initializes a new instance of the **BoxWhisker** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
BoxWhisker

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

gapWidth

Gets or sets a value that determines the width of the gab between groups as a percentage.

The default value for this property is 0.1. The min value is 0 and max value is 1.

Type
number

groupWidth

Gets or sets a value that determines the group width as a percentage.

The default value for this property is 0.8. The min value is 0 and max value is 1.

Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

meanLineStyle

Gets or sets a value that specifies the style for the mean line.

Type
any

meanMarkerStyle

Gets or sets a value that specifies the style for the mean marker.

Type
any

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

quartileCalculation

Gets or sets a value that specifies the quartile calculation method.

Type
`QuartileCalculation`

showInnerPoints

Gets or sets a value that determines whether to show the inner data points for each point in the series.

Type
boolean

showMeanLine

Gets or sets a value that determines whether to show the mean line.

Type
boolean

showMeanMarker

Gets or sets a value that determines whether to show the mean marker.

Type
boolean

showOutliers

Gets or sets a value that determines whether to show outliers.

Outliers are inner points outside the range between the first and third quartiles.

Type
boolean

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

ErrorBar Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
Series
Derived Classes
WjFlexChartErrorBar

Represents an **ErrorBar** series on a **FlexChart**.

The **ErrorBar** series shows error margins and standard deviations at a glance. They can be shown as a standard error amounts, percentages, or standard deviation.

You can also set the error values explicitly to display the exact amounts you want.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- direction
- endStyle
- errorAmount
- errorBarStyle
- hostElement
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- value
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): ErrorBar
```

Initializes a new instance of the **ErrorBar** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
ErrorBar

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart.

Inherited From
`Series`
Type
`ChartType`

collectionView

Gets the `ICollectionView` object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

direction

Gets or sets a value that specifies the direction of the error bars.

Type
`ErrorBarDirection`

endStyle

Gets or sets a value that specifies the end style of the error bars.

Type
`ErrorBarEndStyle`

errorAmount

Gets or sets a value that specifies the meaning of the **`value`** property.

Type
`ErrorAmount`

errorBarStyle

Gets or sets the style used to render the error bars.

Type
any

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

value

Gets or sets a value that specifies the error value of the series.

This property works with the **`errorAmount`** property.

Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

FunctionSeries Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
TrendLineBase
Derived Classes
ParametricFunctionSeries, **YFunctionSeries**

Represents a base class of function series for **FlexChart**.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- max
- min
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FunctionSeries
```

Initializes a new instance of the **FunctionSeries** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
FunctionSeries

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

max

Gets or sets the maximum value of the parameter for calculating a function.

Type
number

min

Gets or sets the minimum value of the parameter for calculating a function.

Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Inherited From
`TrendLineBase`
Returns
number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

MovingAverage Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
TrendLineBase
Derived Classes
WjFlexChartMovingAverage

Represents a moving average trend line for **FlexChart** and **FinancialChart**.

It is a calculation to analyze data points by creating a series of averages of different subsets of the full data set. You may define a different type on each **MovingAverage** object by setting the **type** property on the **MovingAverage** itself.

The **MovingAverage** class has a **period** property that allows you to set the number of periods for computing the average value.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility

Methods

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): MovingAverage
```

Initializes a new instance of the **MovingAverage** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
MovingAverage

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
`SeriesBase`
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the `ICollectionView` object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period of the moving average series. It should be set to integer value greater than 1.

Type
number

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

- **symbolSize**

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

- **symbolStyle**

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

- **type**

Gets or sets the type of the moving average series.

Type
`MovingAverageType`

- **visibility**

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Inherited From
`TrendLineBase`
Returns
number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

ParametricFunctionSeries Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
FunctionSeries
Derived Classes
WjFlexChartParametricFunctionSeries

Represents a parametric function series for **FlexChart**.

The @see::ParametricFunctionSeries plots a function defined by formulas x=f(t) and y=f(t).

The x and y values are calculated by the functions assigned to the **xFunc** and **yFunc** properties.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- max
- min
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- xFunc
- yFunc

Methods

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): ParametricFunctionSeries
```

Initializes a new instance of the **ParametricFunctionSeries** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
ParametricFunctionSeries

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

max

Gets or sets the maximum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

min

Gets or sets the minimum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

xFunc

Gets or sets the function used to calculate the x value.

Type
Function

yFunc

Gets or sets the function used to calculate the y value.

Type
Function

Methods

approximate

```
approximate(value: number): void
```

Gets the approximate x and y from the given value.

Parameters

- **value**: **number**
  The value to calculate.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

TrendLine Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
TrendLineBase
Derived Classes
WjFlexChartTrendLine

Represents a trend line series in a **FlexChart** or **FinancialChart**.

A trend line is a line superimposed on a chart revealing the overall direction of data.

You may define a different fit type for each **TrendLine** series on the **FlexChart** by setting its **fitType** property.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- coefficients
- collectionView
- cssClass
- fitType
- hostElement
- itemsSource
- legendElement
- name
- order
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getEquation
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): TrendLine
```

Initializes a new instance of the **TrendLine** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
TrendLine

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

coefficients

Gets the coefficients of the equation.

Type
number[]

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

fitType

Gets or sets the fit type of the **`TrendLine`**.

Type
`TrendLineFitType`

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **ICollectionView** object that contains the series data.

Inherited From
SeriesBase
Type
any

legendElement

Gets the series element in the legend.

Inherited From
SeriesBase
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
SeriesBase
Type
string

order

Gets or sets the number of terms in a polynomial or Fourier equation.

Set this value to an integer greater than 1. It gets applied when the fitType is set to wijmo.chart.analytics.TrendLineFitType.Polynomial or wijmo.chart.analytics.TrendLineFitType.Fourier.

Type
number

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Returns
number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getEquation

```
getEquation(fmt?: Function): void
```

Gets the formatted equation string for the coefficients.

Parameters

- **fmt**: **Function** OPTIONAL
  The formatting function used to convert the coefficients into strings. This parameter is optional.

Returns
void

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

TrendLineBase Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
SeriesBase
Derived Classes
FunctionSeries, **MovingAverage**, **TrendLine**

Represents base class for various trend lines.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): TrendLineBase
```

Initializes a new instance of the **TrendLineBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
TrendLineBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Returns
number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

Waterfall Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
SeriesBase
Derived Classes
WjFlexChartWaterfall

Represents a Waterfall series of **FlexChart**.

The **Waterfall** series is normally used to demonstrate how the starting position either increases or decreases through a series of changes.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- connectorLines
- cssClass
- hostElement
- intermediateTotalLabels
- intermediateTotalPositions
- itemsSource
- legendElement
- name
- relativeData
- showIntermediateTotal
- showTotal
- start
- startLabel
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- totalLabel
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Waterfall
```

Initializes a new instance of the **Waterfall** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Waterfall

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

connectorLines

Gets or sets a value that determines whether to show connector lines.

Type
boolean

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

intermediateTotalLabels

Gets or sets the name of the property that contains labels for the intermediate total bars. This should be an array or a string.

This property works with the **`showIntermediateTotal`** and **`intermediateTotalPositions`** properties.

Type
any

intermediateTotalPositions

Gets or sets a value of the property that contains the index for positions of the intermediate total bars.

This property works with the **`showIntermediateTotal`** and **`intermediateTotalLabels`** properties.

Type
number[]

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

relativeData

Gets or sets a value that determines whether the given data represents absolute or relative values (differences).

Type
boolean

showIntermediateTotal

Gets or sets a value that determines whether to show intermediate total bars.

This property works with `intermediateTotalPositions` and `intermediateTotalLabels` properties.

Type
boolean

showTotal

Gets or sets a value that determines whether to show the total bar at the end of the chart.

Type
boolean

start

Gets or sets a value that determines the value of the start bar. If start is null, the start bar will not be shown.

Type
number

startLabel

Gets or sets the label of the start bar.

Type
string

style

Gets or sets the series style.

Inherited From
SeriesBase
Type
any

styles

Gets or sets the Waterfall styles.

The following styles are supported:

1. **start**: Specifies the style of the start column.
2. **total**: Specifies the style of the total column.
3. **intermediateTotal**: Specifies the style of the intermediate total column.
4. **falling**: Specifies the style of the falling columns.
5. **rising**: Specifies the style of the rising columns.
6. **connectorLines**: Specifies the style of the connectorLines.

```
waterfall.styles = {
  start: { fill: 'blue', stroke: 'blue' },
  total: { fill: 'yellow', stroke: 'yellow' },
  falling: { fill: 'red', stroke: 'red' },
  rising: { fill: 'green', stroke: 'green' },
  connectorLines: { stroke: 'blue', 'stroke-dasharray': '10, 10' }
}
```

Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

totalLabel

Gets or sets the label of the total bar.

Type
string

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

YFunctionSeries Class

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics
Base Class
FunctionSeries
Derived Classes
WjFlexChartYFunctionSeries

Represents a Y function series of **FlexChart**.

The **YFunctionSeries** plots a function defined by formulas of type y=f(x), specified using the **func** property.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- func
- hostElement
- itemsSource
- legendElement
- max
- min
- name
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- approximate
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): YFunctionSeries
```

Initializes a new instance of the **YFunctionSeries** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
YFunctionSeries

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

func

Gets or sets the function used to calculate Y value.

Type
Function

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

max

Gets or sets the maximum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

min

Gets or sets the minimum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Returns
number

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

ErrorAmount Enum

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Specifies the meaning of the **ErrorBar**'s **value** property.

Members

Name	Value	Description
FixedValue	0	The value property represents the error as an absolute value.
Percentage	1	The value property represents the error as a percentage.
StandardDeviation	2	The value property represents the error as a number of standard deviations.
StandardError	3	The error is the standard error of the mean (value property is not used).
Custom	4	Error values are bound through the **`binding`** property or set to an object with 'plus' and 'minus' values.

ErrorBarDirection Enum

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Specifies the direction of the error bar.

Members

Name	Value	Description
Both	0	Show errors in both directions.
Minus	1	Show errors only in the minus direction.
Plus	2	Show errors only in the plus direction.

ErrorBarEndStyle Enum

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Specifies the end style of the error bars.

Members

Name	Value	Description
Cap	0	Error bars end with a cap.
NoCap	1	Error bars are simple lines.

MovingAverageType Enum

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Specifies the type of MovingAverage Series.

Members

Name	Value	Description
Simple	0	An average of the last n values.
Weighted	1	Weighted average of the last n values, where the weight decreases by 1 with each previous value.
Exponential	2	Weighted average of the last n values, where the weight decreases exponentially with each previous value.
Triangular	3	Weighted average of the last n values, whose result is equivalent to a double smoothed simple moving average.

QuartileCalculation Enum

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Specifies the quartile calculation method of **BoxWhisker** series.

Members

Name	Value	Description
InclusiveMedian	0	Include median value when calculating quartile.
ExclusiveMedian	1	Exclude median value when calculating quartile.

TrendLineFitType Enum

File
`wijmo.chart.analytics.js`
Module
wijmo.chart.analytics

Specifies the fit type for a **TrendLine** series.

Members

Name	Value	Description
Linear	0	A straight line that most closely approximates the data. Y(x) = a * x + b.
Exponential	1	Regression fit to the equation Y(x) = a * exp(b*x).
Logarithmic	2	Regression fit to the equation Y(x) = a * ln(x) + b.
Power	3	Regression fit to the equation Y(x) = a * pow(x, b).
Fourier	4	Regression fit to the equation Y(x) = a + b * cos(x) + c * sin(x) + d * cos(2*x) + e * sin(2*x) + ...
Polynomial	5	Regression fit to the equation Y(x) = a * x^n + b * x^n-1 + c * x^n-2 + ... + z.
MinX	6	The minimum X-value.
MinY	7	The minimum Y-value.
MaxX	8	The maximum X-value.
MaxY	9	The maximum Y-value.
AverageX	10	The average X-value.
AverageY	11	The average Y-value.

wijmo.chart.annotation Module

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation

Defines the **AnnotationLayer** and various annotations for **FlexChart** and **FinancialChart**.

Classes

- AnnotationBase
- AnnotationLayer
- Circle
- Ellipse
- Image
- Line
- Polygon
- Rectangle
- Shape
- Square
- Text

Enums

- AnnotationAttachment
- AnnotationPosition

AnnotationBase Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Derived Classes
Shape, **Text**

Represents the base class of annotations for the **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): AnnotationBase
```

Initializes a new instance of the **AnnotationBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
AnnotationBase

Properties

attachment

Gets or sets the attachment of the annotation.

Type
`AnnotationAttachment`

isVisible

Gets or sets the visibility of the annotation.

Type
boolean

name

Gets or sets the name of the annotation.

Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Type
number

style

Gets or sets the style of the annotation.

Type
any

tooltip

Gets or sets the tooltip of the annotation.

Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Returns
void

AnnotationLayer Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Derived Classes
WjFlexChartAnnotationLayer

Represents an annotation layer for **FlexChart** and **FinancialChart**.

The AnnotationLayer contains a collection of various annotation elements: texts, lines, images, rectangles etc. To use the **AnnotationLayer**, create annotations and push them to the layer's items property.

Constructor

- constructor

Properties

- items

Methods

- getItem
- getItems

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): AnnotationLayer
```

Initializes a new instance of the **`AnnotationLayer`** class.

Parameters

- **chart**: **FlexChartCore**
  A chart to which the **`AnnotationLayer`** is attached.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for **`AnnotationLayer`**.

Returns
AnnotationLayer

Properties

items

Gets the collection of annotation elements in the **`AnnotationLayer`**.

Type
`ObservableArray`

Methods

getItem

```
getItem(name: string): AnnotationBase
```

Gets an annotation element by name in the **`AnnotationLayer`**.

Parameters

- **name**: **string**
  The annotation's name.

Returns
`AnnotationBase`

getItems

```
getItems(name: string): Array
```

Gets the annotation elements by name in the **`AnnotationLayer`**.

Parameters

- **name**: **string**
  The annotations' name.

Returns
Array

Circle Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationCircle

Represents a circle annotation for **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- isVisible
- name
- offset
- point
- pointIndex
- position
- radius
- seriesIndex
- style
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Circle
```

Initializes a new instance of the **Circle** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Circle

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

isVisible

Gets or sets the visibility of the annotation.

Inherited From
AnnotationBase
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

radius

Gets or sets the radius of the **`Circle`** annotation.

Type
number

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

Ellipse Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationEllipse

Represents an ellipse annotation for **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- height
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Ellipse
```

Initializes a new instance of the **Ellipse** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Ellipse

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

height

Gets or sets the height of the **Ellipse** annotation.

Type
number

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

width

Gets or sets the width of the **`Ellipse`** annotation.

Type
number

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

Image Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationImage

Represents an image annotation for **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- height
- href
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Image
```

Initializes a new instance of the **Image** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Image

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

height

Gets or sets the height of the **Image** annotation.

Type
number

href

Gets or sets the href of the **`Image`** annotation.

Type
string

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

width

Gets or sets the width of the **`Image`** annotation.

Type
number

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

Line Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationLine

Represents a line annotation for **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- end
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- start
- style
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Line
```

Initializes a new instance of the **Line** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Line

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

end

Gets or sets the end point of the Line annotation.

Type
DataPoint

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

start

Gets or sets the start point of the **`Line`** annotation.

Type
`DataPoint`

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

Polygon Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationPolygon

Represents a polygon annotation for **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- isVisible
- name
- offset
- point
- pointIndex
- points
- position
- seriesIndex
- style
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Polygon
```

Initializes a new instance of the **Polygon** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Polygon

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

isVisible

Gets or sets the visibility of the annotation.

Inherited From
AnnotationBase
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

points

Gets the collection of points of the **`Polygon`** annotation.

Type
`ObservableArray`

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

Rectangle Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationRectangle

Represents a rectangle annotation for **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- height
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Rectangle
```

Initializes a new instance of the **Rectangle** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Rectangle

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

height

Gets or sets the height of the **Rectangle** annotation.

Type
number

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

width

Gets or sets the width of the `Rectangle` annotation.

Type
number

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

Shape Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
AnnotationBase
Derived Classes
Circle, **Ellipse**, **Image**, **Line**, **Polygon**, **Rectangle**, **Square**

Represents a base class of shape annotations for the **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Shape
```

Initializes a new instance of the **Shape** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Shape

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
`AnnotationBase`
Type
`AnnotationAttachment`

content

Gets or sets the text of the annotation.

Type
string

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

Square Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
Shape
Derived Classes
WjFlexChartAnnotationSquare

Represents a square annotation for the **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- content
- isVisible
- length
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Square
```

Initializes a new instance of the **Square** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Square

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

isVisible

Gets or sets the visibility of the annotation.

Inherited From
AnnotationBase
Type
boolean

length

Gets or sets the length of the **`Square`** annotation.

Type
number

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

Text Class

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation
Base Class
AnnotationBase
Derived Classes
WjFlexChartAnnotationText

Represents a text annotation for the **AnnotationLayer**.

Constructor

- constructor

Properties

- attachment
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- text
- tooltip

Methods

- destroy
- render

Constructor

constructor

```
constructor(options?: any): Text
```

Initializes a new instance of the **Text** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Text

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

isVisible

Gets or sets the visibility of the annotation.

Inherited From
AnnotationBase
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
AnnotationBase
Type
string

offset

Gets or sets the offset of the annotation from the **point**.

Inherited From
AnnotationBase
Type
Point

point

Gets or sets the point of the annotation. The coordinates of points depends on the **attachment** property. See **AnnotationAttachment** for further description.

Inherited From
AnnotationBase
Type
DataPoint

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
AnnotationBase
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **point**.

Inherited From
AnnotationBase
Type
AnnotationPosition

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

text

Gets or sets the text of the annotation.

Type
string

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

Methods

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

AnnotationAttachment Enum

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation

Specifies the attachment of the annotation.

Members

Name	Value	Description
DataIndex	0	Coordinates of the annotation point are defined by the data series index and the data point index.
DataCoordinate	1	Annotation point is specified in data coordinates.
Relative	2	Annotation point is specified as a relative position inside the control where (0,0) is the top left corner and (1,1) is the bottom right corner.
Absolute	3	The annotation point is specified in control's pixel coordinates.

AnnotationPosition Enum

File
`wijmo.chart.annotation.js`
Module
wijmo.chart.annotation

Specifies the position of the annotation.

Members

Name	Value	Description
Center	0	The annotation appears at the Center of the target point.
Top	1	The annotation appears at the Top of the target point.
Bottom	2	The annotation appears at the Bottom of the target point.
Left	4	The annotation appears at the Left of the target point.
Right	8	The annotation appears at the Right of the target point.

wijmo.chart.interaction Module

File
`wijmo.chart.interaction.js`
Module
wijmo.chart.interaction

Defines classes that add interactive features to charts.

Classes

- ChartGestures
- RangeSelector

Enums

- InteractiveAxes
- MouseAction
- Orientation

ChartGestures Class

File
`wijmo.chart.interaction.js`
Module
wijmo.chart.interaction
Derived Classes
WjFlexChartGestures

The **ChartGestures** control allows the user to zoom or pan on the specified **FlexChart**.

To use the **ChartGestures** control, specify the **FlexChart** control on which to zoom or pan.

```
var chartGestures = new wijmo.chart.interaction.ChartGestures(chart);
```

Constructor

- constructor

Properties

- enable
- interactiveAxes
- mouseAction
- posX
- posY
- scaleX
- scaleY

Methods

- remove
- reset

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): ChartGestures
```

Initializes a new instance of the **ChartGestures** class.

Parameters

- **chart**: **FlexChartCore**
  The **FlexChart** that allows the user to zoom or pan.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
ChartGestures

Properties

enable

Gets or sets the enable of the ChartGestures.

Type
boolean

interactiveAxes

Gets or sets the interactive axes of the ChartGestures.

Type
InteractiveAxes

mouseAction

Gets or sets the mouse action of the ChartGestures.

Type
MouseAction

posX

Gets or sets the initial position of the axis X. The value represents initial position on the axis when the Scale is less than 1. Otherwise, the Value has no effect. The Value should lie between 0 to 1.

Type
number

posY

Gets or sets the initial position of the axis Y. The value represents initial position on the axis when the Scale is less than 1. Otherwise, the Value has no effect. The Value should lie between 0 to 1.

Type
number

scaleX

Gets or sets the initial scale of axis X. The scale should be more than 0 and less than or equal to 1. The scale specifies which part of the range between Min and Max is shown. When scale is 1 (default value), the whole axis range is visible.

Type
number

scaleY

Gets or sets the initial scale of axis Y. The scale should be more than 0 and less than or equal to 1. The scale specifies which part of the range between Min and Max is shown. When scale is 1 (default value), the whole axis range is visible.

Type
number

Methods

remove

```
remove(): void
```

Removes the **`ChartGestures`** control from the chart.

Returns
void

reset

```
reset(): void
```

Reset the axis of the chart.

Returns
void

RangeSelector Class

File
`wijmo.chart.interaction.js`
Module
wijmo.chart.interaction
Derived Classes
WjFlexChartRangeSelector

The **RangeSelector** control displays a range selector that allows the user to choose the range of data to display on the specified **FlexChart**.

To use the **RangeSelector** control, specify the **FlexChart** control to display the selected range of data.

The **rangeChanged** event is fired when there is a change in min or max value. For example:

```
var rangeSelector = new wijmo.chart.interaction.RangeSelector(chart);
 rangeSelector.rangeChanged.addHandler(function () {

    // perform related updates
    // e.g. modify displayed range of another chart
    update(rangeSelector.min, rangeSelector.max);
  });
```

Constructor

- constructor

Properties

- isVisible
- max
- maxScale
- min
- minScale
- orientation
- seamless

Methods

- onRangeChanged
- remove

Events

- rangeChanged

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): RangeSelector
```

Initializes a new instance of the **RangeSelector** class.

Parameters

- **chart**: **FlexChartCore**
  The **FlexChart** that displays the selected range.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
RangeSelector

Properties

isVisible

Gets or sets the visibility of the range selector.

Type
boolean

max

Gets or sets the maximum value of the range. If not set, the maximum is calculated automatically.

Type
number

maxScale

Gets or sets the maximum amount of data that can be selected, as a percentage of the total range. This property must be set to a value between zero and one.

Type
number

min

Gets or sets the minimum value of the range. If not set, the minimum is calculated automatically.

Type
number

minScale

Gets or sets the minimum amount of data that can be selected, as a percentage of the overall chart range. This property must be set to a value between zero and one.

Type
number

orientation

Gets or sets the orientation of the range selector.

Type
`Orientation`

seamless

Gets or sets a value that determines whether the min/max elements may be reversed by dragging one over the other.

Type
boolean

Methods

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

Raises the **rangeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

remove

```
remove(): void
```

Removes the **RangeSelector** control from the chart.

Returns
void

Events

rangeChanged

Occurs after the range changes.

Arguments
EventArgs

InteractiveAxes Enum

File
`wijmo.chart.interaction.js`
Module
wijmo.chart.interaction

Specifies the interactive axes of the chart gestures.

Members

Name	Value	Description
X	0	Interactive Axis X.
Y	1	Interactive Axis Y.
XY	2	Interactive Both Axis X and Axis Y.

MouseAction Enum

File
`wijmo.chart.interaction.js`
Module
wijmo.chart.interaction

Specifies the mouse action of the chart gestures.

Members

Name	Value	Description
Zoom	0	Zoom chart by mouse.
Pan	1	Pan chart by mouse.

Orientation Enum

File
`wijmo.chart.interaction.js`
Module
wijmo.chart.interaction

Specifies the orientation of the range selector.

Members

Name	Value	Description
X	0	Horizontal, x-data range.
Y	1	Vertical, y-data range.

wijmo.chart.animation Module

File
`wijmo.chart.animation.js`
Module
wijmo.chart.animation

Defines the **ChartAnimation** for **FlexChart**, **FinancialChart** and **FlexPie**.

Classes

- ChartAnimation

Enums

- AnimationMode
- Easing

ChartAnimation Class

File
`wijmo.chart.animation.js`
Module
wijmo.chart.animation
Derived Classes
WjFlexChartAnimation

Represents the animation for **FlexChart**, **FinancialChart** and **FlexPie**.

The **ChartAnimation** provides built-in animation while loading and updating the chart. The animation can be configured by the user through several properties that include duration, easing function, animation mode.

Constructor

- constructor

Properties

- animationMode
- axisAnimation
- duration
- easing

Methods

- animate

Constructor

constructor

```
constructor(chart: FlexChartBase, options?: any): ChartAnimation
```

Initializes a new instance of the **ChartAnimation** class.

Parameters

- **chart**: **FlexChartBase**
  A chart to which the **ChartAnimation** is attached.

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data for **ChartAnimation**.

Returns
ChartAnimation

Properties

animationMode

Gets or sets whether the plot points animate one at a time, series by series, or all at once. The whole animation is still completed within the duration.

Type
AnimationMode

axisAnimation

Gets or sets a value indicating whether animation is applied to the axis.

Type
boolean

duration

Gets or sets the length of entire animation in milliseconds.

Type
number

easing

Gets or sets the easing function applied to the animation.

Type
`Easing`

Methods

animate

```
animate(): void
```

Performs the animation.

Returns
void

AnimationMode Enum

File
`wijmo.chart.animation.js`
Module
wijmo.chart.animation

Specifies the animation mode whether chart should animate one point at a time, series by series, or all at once.

Members

Name	Value	Description
All	0	All points and series are animated at once.
Point	1	Animation is performed point by point. Multiple series are animated simultaneously at the same time.
Series	2	Animation is performed series by series. Entire series is animated at once, following the same animation as the "All" option, but just one series at a time.

Easing Enum

File
`wijmo.chart.animation.js`
Module
wijmo.chart.animation

Specifies the rate of change of a parameter over time.

Members

Name	Value	Description
Linear	0	Simple linear tweening, no easing and no acceleration.
Swing	1	Easing equation for a swing easing
EaseInQuad	2	Easing equation for a quadratic easing in, accelerating from zero velocity.
EaseOutQuad	3	Easing equation for a quadratic easing out, decelerating to zero velocity.
EaseInOutQuad	4	Easing equation for a quadratic easing in and out, acceleration until halfway, then deceleration.
EaseInCubic	5	Easing equation for a cubic easing in - accelerating from zero velocity.
EaseOutCubic	6	Easing equation for a cubic easing out - decelerating to zero velocity.
EaseInOutCubic	7	Easing equation for a cubic easing in and out - acceleration until halfway, then deceleration.
EaseInQuart	8	Easing equation for a quartic easing in - accelerating from zero velocity.
EaseOutQuart	9	Easing equation for a quartic easing out - decelerating to zero velocity.
EaseInOutQuart	10	Easing equation for a quartic easing in and out - acceleration until halfway, then deceleration.
EaseInQuint	11	Easing equation for a quintic easing in - accelerating from zero velocity.
EaseOutQuint	12	Easing equation for a quintic easing out - decelerating to zero velocity.
EaseInOutQuint	13	Easing equation for a quintic easing in and out - acceleration until halfway, then deceleration.
EaseInSine	14	Easing equation for a sinusoidal easing in - accelerating from zero velocity.
EaseOutSine	15	Easing equation for a sinusoidal easing out - decelerating to zero velocity.
EaseInOutSine	16	Easing equation for a sinusoidal easing in and out - acceleration until halfway, then deceleration.
EaseInExpo	17	Easing equation for an exponential easing in - accelerating from zero velocity.
EaseOutExpo	18	Easing equation for an exponential easing out - decelerating to zero velocity.
EaseInOutExpo	19	Easing equation for an exponential easing in and out - acceleration until halfway, then deceleration.
EaseInCirc	20	Easing equation for a circular easing in - accelerating from zero velocity.
EaseOutCirc	21	Easing equation for a circular easing out - decelerating to zero velocity.
EaseInOutCirc	22	Easing equation for a circular easing in and out - acceleration until halfway, then deceleration.
EaseInBack	23	Easing equation for a back easing in - accelerating from zero velocity.
EaseOutBack	24	Easing equation for a back easing out - decelerating to zero velocity.
EaseInOutBack	25	Easing equation for a back easing in and out - acceleration until halfway, then deceleration.
EaseInBounce	26	Easing equation for a bounce easing in - accelerating from zero velocity.
EaseOutBounce	27	Easing equation for a bounce easing out - decelerating to zero velocity.
EaseInOutBounce	28	Easing equation for a bounce easing in and out - acceleration until halfway, then deceleration.
EaseInElastic	29	Easing equation for an elastic easing in - accelerating from zero velocity.
EaseOutElastic	30	Easing equation for an elastic easing out - decelerating to zero velocity.
EaseInOutElastic	31	Easing equation for an elastic easing in and out - acceleration until halfway, then deceleration.

wijmo.chart.render Module

File
`wijmo.chart.render.js`
Module
wijmo.chart.render

Provides various render engines to render wijmo chart. Add this module on page to support chart export in IE browsers.

wijmo.chart.hierarchical Module

File
`wijmo.chart.hierarchical.js`
Module
wijmo.chart.hierarchical

Defines the **Sunburst** chart control and its associated classes.

Classes

- Sunburst
- TreeMap

Enums

- TreeMapType

Sunburst Class

File
`wijmo.chart.hierarchical.js`
Module
wijmo.chart.hierarchical
Base Class
FlexPie
Derived Classes
WjSunburst

Sunburst chart control.

Constructor

- constructor

Properties

- binding
- bindingName
- childItemsPath
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- innerRadius
- isAnimated
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- offset
- palette
- plotMargin
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition
- selectionMode
- startAngle
- tooltip

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged

Constructor

constructor

```
constructor(element: any, options?): FlexPie
```

Initializes a new instance of the **FlexPie** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A Javascript object containing initialization data for the control.

Inherited From
FlexPie
Returns
FlexPie

Properties

binding

Gets or sets the name of the property that contains the chart values.

Inherited From
FlexPie
Type
string

bindingName

Gets or sets the name of the property containing name of the data item; it should be an array or a string.

Type
any

childItemsPath

Gets or sets the name of the property (or properties) used to generate child items in hierarchical data.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

Set this property to an array containing the names of the properties that contain child items at each level, when the items are child items at different levels with different names (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Type
any

collectionView

Gets the **`ICollectionView`** object that contains the chart data.

Inherited From
`FlexChartBase`
Type
`ICollectionView`

dataLabel

Gets or sets the point data label.

Inherited From
`FlexPie`
Type
`PieDataLabel`

footer

Gets or sets the text displayed in the chart footer.

Inherited From
`FlexChartBase`
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
`FlexChartBase`
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

innerRadius

Gets or sets the size of the pie's inner radius.

The inner radius is measured as a fraction of the pie radius.

The default value for this property is zero, which creates a pie. Setting this property to values greater than zero creates pies with a hole in the middle, also known as doughnut charts.

Inherited From
`FlexPie`
Type
number

isAnimated

Gets or sets a value indicating whether to use animation when items are selected.

See also the `selectedItemPosition` and `selectionMode` properties.

Inherited From
`FlexPie`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

offset

Gets or sets the offset of the slices from the pie center.

The offset is measured as a fraction of the pie radius.

Inherited From
`FlexPie`
Type
number

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **`Palettes`** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
`FlexChartBase`
Type
string[]

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

reversed

Gets or sets a value that determines whether angles are reversed (counter-clockwise).

The default value is false, which causes angles to be measured in the clockwise direction.

Inherited From
`FlexPie`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the selected slice.

Inherited From
`FlexPie`
Type
number

selectedItemOffset

Gets or sets the offset of the selected slice from the pie center.

Offsets are measured as a fraction of the pie radius.

Inherited From
`FlexPie`
Type
number

selectedItemPosition

Gets or sets the position of the selected slice.

Setting this property to a value other than 'None' causes the pie to rotate when an item is selected.

Note that in order to select slices by clicking the chart, you must set the **`selectionMode`** property to "Point".

Inherited From
`FlexPie`
Type
`Position`

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

startAngle

Gets or sets the starting angle for the pie slices, in degrees.

Angles are measured clockwise, starting at the 9 o'clock position.

Inherited From
`FlexPie`
Type
number

tooltip

Gets the chart's `Tooltip`.

Inherited From
`FlexPie`
Type
`ChartTooltip`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexPie`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
FlexChartBase
Returns
Point

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

TreeMap Class

File
`wijmo.chart.hierarchical.js`
Module
wijmo.chart.hierarchical
Base Class
FlexChartBase
Derived Classes
WjTreeMap

The **TreeMap** control displays hierarchical (tree-structured) data as a set of nested rectangles. Each branch of the tree is given a rectangle, which is then tiled with smaller rectangles representing sub-branches. A leaf node's rectangle has an area proportional to a specified dimension of the data. Often the leaf nodes are colored to show a separate dimension of the data.

To use the **TreeMap** control, set the **itemsSource** property to an array containing the data and use the **binding** and **bindingName** properties to set the properties that contain the item values and names.

Constructor

- constructor

Properties

- binding
- bindingName
- childItemsPath
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- maxDepth
- palette
- plotMargin
- rightToLeft
- selectionMode
- tooltip
- type

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged

Constructor

constructor

```
constructor(element: any, options?): TreeMap
```

Initializes a new instance of the **TreeMap** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A Javascript object containing initialization data for the control.

Returns
TreeMap

Properties

binding

Gets or sets the name of the property of the data item that contains the chart value.

The binding property is used to calculate the size of the node as compared to other node values. The property should contain numeric data.

Type
string

bindingName

Gets or sets the name of the property containing name of the data item. The bindingName property is used to show name of the node. It should be an array or a string.

Type
any

childItemsPath

Gets or sets the name of the property (or properties) used to generate child items in hierarchical data.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

Set this property to an array containing the names of the properties that contain child items at each level, when the items are child items at different levels with different names (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Type
any

collectionView

Gets the **`ICollectionView`** object that contains the chart data.

Inherited From
`FlexChartBase`
Type
`ICollectionView`

dataLabel

Gets or sets the **`DataLabel`** of the treemap.

Type
`DataLabel`

footer

Gets or sets the text displayed in the chart footer.

Inherited From
`FlexChartBase`
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
`FlexChartBase`
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

maxDepth

Gets or sets the maximum number of node levels to show in the current view. These levels are flattened into the current plane. If a treemap has more levels than this value, user has to move up and down.

Type
number

palette

Gets or sets an array of default colors to be used in a treemap.

The array contains strings that represent CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

Or contains titleColor, maxColor, minColor separately. For example:

```
chart.palette = [{
     titleColor: '#00277d',
     maxColor: 'rgba(0,39,125,0.7)',
     minColor: 'rgba(168,187,230,0.7)'
 }, {
     titleColor: '#7d1f00',
     maxColor: 'rgba(125,21,0,0.7)',
     minColor: 'rgba(230,183,168,0.7)'
 }, {
     titleColor: '#007d27',
     maxColor: 'rgba(0,125,39,0.7)',
     minColor: 'rgba(168,230,188,0.7)'
 }];
```

Type
string[]

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

tooltip

Gets the chart's **`Tooltip`**.

Type
`ChartTooltip`

type

Gets or sets the **`TreeMapType`** of the treemap.

Type
`TreeMapType`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
FlexChartBase
Returns
Point

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

TreeMapType Enum

File
`wijmo.chart.hierarchical.js`
Module
wijmo.chart.hierarchical

Specifies the treemap type.

Members

Name	Value	Description
Squarified	0	Shows squarified treemap.
Horizontal	1	Shows horizontal squarified treemap.
Vertical	2	Shows vertical squarified treemap.

wijmo.chart.radar Module

File
`wijmo.chart.radar.js`
Module
wijmo.chart.radar

Defines the **FlexRadar** control and its associated classes.

Classes

- FlexRadar
- FlexRadarAxis
- FlexRadarSeries

Enums

- RadarChartType

FlexRadar Class

File
`wijmo.chart.radar.js`
Module
wijmo.chart.radar
Base Class
FlexChartCore
Derived Classes
WjFlexRadar

radar chart control.

Constructor

- constructor

Properties

- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- palette
- plotAreas
- plotMargin
- reversed
- rightToLeft
- selection
- selectionMode
- series
- stacking
- startAngle
- symbolSize
- tooltip
- totalAngle

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

Constructor

constructor

```
constructor(element: any, options?): FlexRadar
```

Initializes a new instance of the **FlexRadar** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
FlexRadar

Properties

axes

Gets the collection of **Axis** objects.

Inherited From
FlexChartCore
Type
ObservableArray

axisX

Gets or sets the main X axis.

Inherited From
FlexChartCore
Type
Axis

axisY

Gets or sets the main Y axis.

Inherited From
`FlexChartCore`
Type
`Axis`

binding

Gets or sets the name of the property that contains the Y values.

Inherited From
`FlexChartCore`
Type
string

bindingX

Gets or sets the name of the property that contains the X data values.

Inherited From
`FlexChartCore`
Type
string

chartType

Gets or sets the type of radar chart to be created.

Type
`RadarChartType`

collectionView

Gets the **ICollectionView** object that contains the chart data.

Inherited From
FlexChartBase
Type
ICollectionView

dataLabel

Gets or sets the point data label.

Inherited From
FlexChartCore
Type
DataLabel

footer

Gets or sets the text displayed in the chart footer.

Inherited From
FlexChartBase
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
FlexChartBase
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

interpolateNulls

Gets or sets a value that determines whether to interpolate null values in the data.

If true, the chart interpolates the value of any missing data based on neighboring points. If false, it leaves a break in lines and areas at the points with null values.

Inherited From
`FlexChartCore`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

legendToggle

Gets or sets a value indicating whether clicking legend items toggles the series visibility in the chart.

Inherited From
`FlexChartCore`
Type
boolean

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **Palettes** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
FlexChartBase
Type
string[]

plotAreas

Gets the collection of **PlotArea** objects.

Inherited From
FlexChartCore
Type
PlotAreaCollection

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

reversed

Gets or sets a value that determines whether angles are reversed (counter-clockwise).

The default value is false, which causes angles to be measured in the clockwise direction.

Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selection

Gets or sets the selected chart series.

Inherited From
`FlexChartCore`
Type
`SeriesBase`

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

series

Gets the collection of **`Series`** objects.

Inherited From
`FlexChartCore`
Type
`ObservableArray`

stacking

Gets or sets a value that determines whether and how the series objects are stacked.

Type
`Stacking`

startAngle

Gets or sets the starting angle for the radar, in degrees.

Angles are measured clockwise, starting at the 12 o'clock position.

Type
number

symbolSize

Gets or sets the size of the symbols used for all Series objects in this **`FlexChart`**.

This property may be overridden by the symbolSize property on each **`Series`** object.

Inherited From
`FlexChartCore`
Type
number

tooltip

Gets the chart **Tooltip** object.

The tooltip content is generated using a template that may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

To modify the template, assign a new value to the tooltip's content property. For example:

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

You can disable chart tooltips by setting the template to an empty string.

You can also use the **tooltip** property to customize tooltip parameters such as **showDelay** and **hideDelay**:

```
chart.tooltip.showDelay = 1000;
```

See **ChartTooltip** properties for more details and options.

Inherited From
FlexChartCore
Type
ChartTooltip

totalAngle

Gets or sets the total angle for the radar, in degrees. Its default value is 360. The value must be greater than 0, or less than or equal to 360.

Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Converts a **Point** from data coordinates to control coordinates.

Parameters

- **pt**: **any**
  Point in data coordinates, or X coordinate of a point in data coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Inherited From
FlexChartCore
Returns
Point

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

Raises the **seriesVisibilityChanged** event.

Parameters

- **e**: **SeriesEventArgs**
  The **SeriesEventArgs** object that contains the event data.

Inherited From
FlexChartCore
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
`FlexChartBase`
Returns
`Point`

pointToData

```
pointToData(pt: any, y?: number): Point
```

Converts a **`Point`** from control coordinates to chart data coordinates.

Parameters

- **pt**: **any**
  The point to convert, in control coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

seriesVisibilityChanged

Occurs when the series visibility changes, for example when the legendToggle property is set to true and the user clicks the legend.

Inherited From
`FlexChartCore`
Arguments
`SeriesEventArgs`

FlexRadarAxis Class

File
`wijmo.chart.radar.js`
Module
wijmo.chart.radar
Base Class
Axis
Derived Classes
WjFlexRadarAxis

Represents an axis in the radar chart.

Constructor

- constructor

Properties

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- reversed
- title

Methods

- convert
- convertBack
- onRangeChanged

Events

- rangeChanged

Constructor

constructor

```
constructor(position?: Position): Axis
```

Initializes a new instance of the **Axis** class.

Parameters

- **position**: **Position** OPTIONAL
  The position of the axis on the chart.

Inherited From
Axis
Returns
Axis

Properties

actualMax

Gets the actual axis maximum.

It returns a number or a Date object (for time-based data).

Inherited From
Axis
Type
any

actualMin

Gets the actual axis minimum.

It returns a number or a Date object (for time-based data).

Inherited From
Axis
Type
any

axisLine

Gets or sets a value indicating whether the axis line is visible.

Inherited From
`Axis`
Type
boolean

axisType

Gets the axis type.

Inherited From
`Axis`
Type
`AxisType`

binding

Gets or sets the comma-separated property names for the **`itemsSource`** property to use in axis labels.

The first name specifies the value on the axis, the second represents the corresponding axis label. The default value is 'value,text'.

Inherited From
`Axis`
Type
string

format

Gets or sets the format string used for the axis labels (see **`Globalize`**).

Inherited From
`Axis`
Type
string

hostElement

Gets the axis host element.

Inherited From
`Axis`
Type
SVGGElement

itemFormatter

Gets or sets the itemFormatter function for the axis labels.

If specified, the function takes two parameters:

- **render engine**: The **IRenderEngine** object to be used in formatting the labels.
- **current label**: An object with the following properties:
  - **value**: The value of the axis label to format.
  - **text**: The text to use in the label.
  - **pos**: The position in control coordinates at which the label is to be rendered.
  - **cls**: The CSS class to be applied to the label.

The function returns the label parameters of labels for which properties are modified.

For example:

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // red text
        label.cls = null; // no default CSS
     }
    return label;
}
```

Inherited From
`Axis`
Type
Function

itemsSource

Gets or sets the items source for the axis labels.

Names of the properties are specified by the **`binding`** property.

For example:

```
// default value for Axis.binding is 'value,text'
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

Inherited From
`Axis`
Type
any

labelAlign

Gets or sets the label alignment.

By default the labels are centered. The supported values are 'left' and 'right for x-axis and 'top' and 'bottom' for y-axis.

Inherited From
`Axis`
Type
string

labelAngle

Gets or sets the rotation angle of the axis labels.

The angle is measured in degrees with valid values ranging from -90 to 90.

Inherited From
`Axis`
Type
number

labelPadding

Gets or sets the label padding.

Inherited From
`Axis`
Type
number

labels

Gets or sets a value indicating whether the axis labels are visible.

Inherited From
`Axis`
Type
boolean

logBase

Gets or sets the logarithmic base of the axis.

If the base is not specified the axis uses a linear scale.

Use the **`logBase`** property to spread data that is clustered around the origin. This is common in several financial and economic data sets.

Inherited From
`Axis`
Type
number

majorGrid

Gets or sets a value indicating whether the axis includes grid lines.

Inherited From
`Axis`
Type
boolean

majorTickMarks

Gets or sets the location of the axis tick marks.

Inherited From
`Axis`
Type
`TickMark`

majorUnit

Gets or sets the number of units between axis labels.

If the axis contains date values, then the units are expressed in days.

Inherited From
`Axis`
Type
number

max

Gets or sets the maximum value shown on the axis.

If not set, the maximum is calculated automatically. The value can be a number or a Date object (for time-based data).

Inherited From
`Axis`
Type
any

min

Gets or sets the minimum value shown on the axis.

If not set, the minimum is calculated automatically. The value can be a number or a Date object (for time-based data).

Inherited From
`Axis`
Type
any

minorGrid

Gets or sets a value indicating whether the axis includes minor grid lines.

Inherited From
`Axis`
Type
boolean

minorTickMarks

Gets or sets the location of the minor axis tick marks.

Inherited From
`Axis`
Type
`TickMark`

minorUnit

Gets or sets the number of units between minor axis ticks.

If the axis contains date values, then the units are expressed in days.

Inherited From
`Axis`
Type
number

name

Gets or sets the axis name.

Inherited From
`Axis`
Type
string

origin

Gets or sets the value at which an axis crosses the perpendicular axis.

Inherited From
`Axis`
Type
number

overlappingLabels

Gets or sets a value indicating how to handle the overlapping axis labels.

Inherited From
`Axis`
Type
`OverlappingLabels`

plotArea

Gets or sets the plot area for the axis.

Inherited From
`Axis`
Type
`PlotArea`

position

Gets or sets the position of the axis with respect to the plot area.

Inherited From
`Axis`
Type
`Position`

reversed

Gets or sets a value indicating whether the axis is reversed (top to bottom or right to left).

Inherited From
`Axis`
Type
boolean

title

Gets or sets the title text shown next to the axis.

Inherited From
`Axis`
Type
string

Methods

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

Converts the specified value from data to pixel coordinates.

Parameters

- **val**: **number**
  The data value to convert.
- **maxValue**: **number** OPTIONAL
  The max value of the data, it's optional.
- **minValue**: **number** OPTIONAL
  The min value of the data, it's optional.

Inherited From
`Axis`
Returns
number

convertBack

```
convertBack(val: number): number
```

Converts the specified value from pixel to data coordinates.

Parameters

- **val**: **number**
  The pixel coordinates to convert back.

Inherited From
Axis
Returns
number

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

Raises the **rangeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Axis
Returns
void

Events

rangeChanged

Occurs when the axis range changes.

Inherited From
Axis
Arguments
EventArgs

FlexRadarSeries Class

File
`wijmo.chart.radar.js`
Module
wijmo.chart.radar
Base Class
SeriesBase
Derived Classes
WjFlexRadarSeries

Represents a series of data points to display in the chart.

The **FlexRadarSeries** class supports all basic chart types. You may define a different chart type on each **FlexRadarSeries** object that you add to the **FlexRadar** series collection. This overrides the **chartType** property set on the chart that is the default for all **FlexRadarSeries** objects in its collection.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SeriesBase
```

Initializes a new instance of the **SeriesBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
SeriesBase
Returns
SeriesBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart. Please note that ColumnVolume, EquiVolume, CandleVolume and ArmsCandleVolume chart types are not supported and should be set on the **`FinancialChart`**.

Type
`RadarChartType`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

RadarChartType Enum

File
`wijmo.chart.radar.js`
Module
wijmo.chart.radar

Specifies the type of radar chart.

Members

Name	Value	Description
Column	0	Shows vertical bars and allows you to compare values of items across categories.
Scatter	1	Shows patterns within the data using X and Y coordinates.
Line	2	Shows trends over a period of time or across categories.
LineSymbols	3	Shows line chart with a symbol on each data point.
Area	4	Shows line chart with the area below the line filled with color.

wijmo.gauge Module

File
`wijmo.gauge.js`
Module
wijmo.gauge

Defines the **RadialGauge**, **LinearGauge**, and **BulletGraph** controls.

Unlike many gauge controls, Wijmo gauges concentrate on the data being displayed, with little extraneous color and markup elements. They were designed to be easy to use and to read, especially on small-screen devices.

Wijmo gauges are composed of **Range** objects. Every Wijmo gauge has at least two ranges: the "face" and the "pointer".

- The "face" represents the gauge background. The "min" and "max" properties of the face range correspond to the "min" and "max" properties of the gauge control, and limit the values that the gauge can display.
- The "pointer" is the range that indicates the gauge's current value. The "max" property of the pointer range corresponds to the "value" property of the gauge.

In addition to these two special ranges, gauges may have any number of additional ranges added to their "ranges" collection. These additional ranges can be used for two things:

- By default, the extra ranges appear as part of the gauge background. This way you can show 'zones' within the gauge, like 'good,' 'average,' and 'bad' for example.
- If you set the gauge's "showRanges" property to false, the additional ranges are not shown. Instead, they are used to automatically set the color of the "pointer" based on the current value.

Classes

- BulletGraph
- Gauge
- LinearGauge
- RadialGauge
- Range

Enums

- GaugeDirection
- ShowText

BulletGraph Class

File
`wijmo.gauge.js`
Module
wijmo.gauge
Base Class
LinearGauge
Derived Classes
WjBulletGraph

The **BulletGraph** is a type of linear gauge designed specifically for use in dashboards. It displays a single key measure along with a comparative measure and qualitative ranges to instantly signal whether the measure is good, bad, or in some other state.

Bullet Graphs were created and popularized by dashboard design expert Stephen Few. You can find more details and examples on **Wikipedia**.

Example

Show me (http://jsfiddle.net/Wijmo5/vqrwdvgq)

Constructor

- constructor

Properties

- bad
- controlTemplate
- direction
- face
- format
- getText
- good
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- step
- target
- thickness
- thumbSize
- tickSpacing
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): BulletGraph
```

Initializes a new instance of the **BulletGraph** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
BulletGraph

Properties

bad

Gets or sets a reference value considered bad for the measure.

Type
number

STATIC controlTemplate

Gets or sets the template used to instantiate **Gauge** controls.

Inherited From
Gauge
Type
any

direction

Gets or sets the direction in which the gauge is filled.

Inherited From
LinearGauge
Type
GaugeDirection

face

Gets or sets the **Range** used to represent the gauge's overall geometry and appearance.

Inherited From
Gauge
Type
Range

format

Gets or sets the format string used to display gauge values as text.

Inherited From
Gauge
Type
string

getText

Gets or sets a callback that returns customized strings used to display gauge values.

Use this property if you want to customize the strings shown on the gauge in cases where the **`format`** property is not enough.

If provided, the callback should be a function as that takes as parameters the gauge, the part name, the value, and the formatted value. The callback should return the string to be displayed on the gauge.

For example:

```
// callback to convert values into strings
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return 'Empty!';
      if (value <= 25) return 'Low...';
      if (value <= 95) return 'Good';
      return 'Full';
    case 'min':
      return 'EMPTY';
    case 'max':
      return 'FULL';
  }
  return text;
}
```

Inherited From
Gauge
Type
Function

good

Gets or sets a reference value considered good for the measure.

Type
number

hasShadow

Gets or sets a value that indicates whether the gauge displays a shadow effect.

Inherited From
Gauge
Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isAnimated

Gets or sets a value that indicates whether the gauge animates value changes.

Inherited From
`Gauge`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can edit the value using the mouse and keyboard.

Inherited From
`Gauge`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

max

Gets or sets the maximum value that can be displayed on the gauge.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Gauge`
Type
number

min

Gets or sets the minimum value that can be displayed on the gauge.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Gauge`
Type
number

origin

Gets or sets the starting point used for painting the range.

By default, this property is set to null, which causes the value range to start at the gauge's minimum value, or zero if the minimum is less than zero.

Inherited From
`Gauge`
Type
number

pointer

Gets or sets the **`Range`** used to represent the gauge's current value.

Inherited From
`Gauge`
Type
`Range`

ranges

Gets the collection of ranges in this gauge.

Inherited From
`Gauge`
Type
`ObservableArray`

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showRanges

Gets or sets a value that indicates whether the gauge displays the ranges contained in the **ranges** property.

If this property is set to false, the ranges contained in the **ranges** property are not displayed in the gauge. Instead, they are used to interpolate the color of the **pointer** range while animating value changes.

Inherited From
Gauge
Type
boolean

showText

Gets or sets the **ShowText** values to display as text in the gauge.

Inherited From
Gauge
Type
ShowText

showTicks

Gets or sets a property that determines whether the gauge should display tickmarks at each **step** value.

The tickmarks can be formatted in CSS using the **wj-gauge** and **wj-ticks** class names. For example:

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

Inherited From
Gauge
Type
boolean

step

Gets or sets the amount to add to or subtract from the **`value`** property when the user presses the arrow keys or moves the mouse wheel.

Inherited From
Gauge
Type
number

target

Gets or sets the target value for the measure.

Type
number

thickness

Gets or sets the thickness of the gauge, on a scale between zero and one.

Setting the thickness to one causes the gauge to fill as much of the control area as possible. Smaller values create thinner gauges.

Inherited From
Gauge
Type
number

thumbSize

Gets or sets the size of the element that shows the gauge's current value, in pixels.

Inherited From
Gauge
Type
number

tickSpacing

Gets or sets the spacing between tickmarks.

Set the **`showTicks`** property to true if you want the gauge to show tickmarks along its face. By default, the interval between tickmarks is defined by the **`step`** property.

Use the **`tickSpacing`** property to override the default and use a spacing that is different from the **`step`** value. Set the **`tickSpacing`** property to null to revert to the default behavior.

Inherited From
Gauge
Type
number

value

Gets or sets the value displayed on the gauge.

Inherited From
Gauge
Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): number
```

Gets a number that corresponds to the value of the gauge at a given point.

For example:

```
// hit test a point when the user clicks on the gauge
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates, or a MouseEvent object, or the x coordinate of the point.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
Gauge
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Gauge
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Indicates whether to update the control layout as well as the content.

Inherited From
Gauge
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes.

Inherited From
`Gauge`
Arguments
`EventArgs`

Gauge Class

File
`wijmo.gauge.js`
Module
wijmo.gauge
Base Class
Control
Derived Classes
LinearGauge, **RadialGauge**

Base class for the Wijmo Gauge controls (abstract).

Constructor

- constructor

Properties

- controlTemplate
- face
- format
- getText
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- step
- thickness
- thumbSize
- tickSpacing
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): Gauge
```

Initializes a new instance of the **Gauge** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
Gauge

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **Gauge** controls.

Type
any

face

Gets or sets the **Range** used to represent the gauge's overall geometry and appearance.

Type
Range

format

Gets or sets the format string used to display gauge values as text.

Type
string

getText

Gets or sets a callback that returns customized strings used to display gauge values.

Use this property if you want to customize the strings shown on the gauge in cases where the **`format`** property is not enough.

If provided, the callback should be a function as that takes as parameters the gauge, the part name, the value, and the formatted value. The callback should return the string to be displayed on the gauge.

For example:

```
// callback to convert values into strings
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return 'Empty!';
      if (value <= 25) return 'Low...';
      if (value <= 95) return 'Good';
      return 'Full';
    case 'min':
      return 'EMPTY';
    case 'max':
      return 'FULL';
  }
  return text;
}
```

Type
Function

hasShadow

Gets or sets a value that indicates whether the gauge displays a shadow effect.

Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isAnimated

Gets or sets a value that indicates whether the gauge animates value changes.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can edit the value using the mouse and keyboard.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

max

Gets or sets the maximum value that can be displayed on the gauge.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Type
number

min

Gets or sets the minimum value that can be displayed on the gauge.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Type
number

origin

Gets or sets the starting point used for painting the range.

By default, this property is set to null, which causes the value range to start at the gauge's minimum value, or zero if the minimum is less than zero.

Type
number

pointer

Gets or sets the **Range** used to represent the gauge's current value.

Type
Range

ranges

Gets the collection of ranges in this gauge.

Type
ObservableArray

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showRanges

Gets or sets a value that indicates whether the gauge displays the ranges contained in the **`ranges`** property.

If this property is set to false, the ranges contained in the **`ranges`** property are not displayed in the gauge. Instead, they are used to interpolate the color of the **`pointer`** range while animating value changes.

Type
boolean

showText

Gets or sets the **`ShowText`** values to display as text in the gauge.

Type
`ShowText`

showTicks

Gets or sets a property that determines whether the gauge should display tickmarks at each **`step`** value.

The tickmarks can be formatted in CSS using the **wj-gauge** and **wj-ticks** class names. For example:

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

Type
boolean

step

Gets or sets the amount to add to or subtract from the **value** property when the user presses the arrow keys or moves the mouse wheel.

Type
number

thickness

Gets or sets the thickness of the gauge, on a scale between zero and one.

Setting the thickness to one causes the gauge to fill as much of the control area as possible. Smaller values create thinner gauges.

Type
number

thumbSize

Gets or sets the size of the element that shows the gauge's current value, in pixels.

Type
number

tickSpacing

Gets or sets the spacing between tickmarks.

Set the **showTicks** property to true if you want the gauge to show tickmarks along its face. By default, the interval between tickmarks is defined by the **step** property.

Use the **tickSpacing** property to override the default and use a spacing that is different from the **step** value. Set the **tickSpacing** property to null to revert to the default behavior.

Type
number

value

Gets or sets the value displayed on the gauge.

Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): number
```

Gets a number that corresponds to the value of the gauge at a given point.

For example:

```
// hit test a point when the user clicks on the gauge
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates, or a MouseEvent object, or the x coordinate of the point.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Indicates whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes.

Arguments
`EventArgs`

LinearGauge Class

File
`wijmo.gauge.js`
Module
wijmo.gauge
Base Class
Gauge
Derived Classes
BulletGraph, **WjLinearGauge**

The **LinearGauge** displays a linear scale with an indicator that represents a single value and optional ranges to represent reference values.

If you set the gauge's **isReadOnly** property to false, then users will be able to edit the value by clicking on the gauge.

Example

Show me (http://jsfiddle.net/Wijmo5/wkcehhvu)

Constructor

- constructor

Properties

- controlTemplate
- direction
- face
- format
- getText
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- step
- thickness
- thumbSize
- tickSpacing
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): LinearGauge
```

Initializes a new instance of the **LinearGauge** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
LinearGauge

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **Gauge** controls.

Inherited From
Gauge
Type
any

direction

Gets or sets the direction in which the gauge is filled.

Type
GaugeDirection

face

Gets or sets the **Range** used to represent the gauge's overall geometry and appearance.

Inherited From
Gauge
Type
Range

format

Gets or sets the format string used to display gauge values as text.

Inherited From
Gauge
Type
string

getText

Gets or sets a callback that returns customized strings used to display gauge values.

Use this property if you want to customize the strings shown on the gauge in cases where the **`format`** property is not enough.

If provided, the callback should be a function as that takes as parameters the gauge, the part name, the value, and the formatted value. The callback should return the string to be displayed on the gauge.

For example:

```
// callback to convert values into strings
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return 'Empty!';
      if (value <= 25) return 'Low...';
      if (value <= 95) return 'Good';
      return 'Full';
    case 'min':
      return 'EMPTY';
    case 'max':
      return 'FULL';
  }
  return text;
}
```

Inherited From
Gauge
Type
Function

hasShadow

Gets or sets a value that indicates whether the gauge displays a shadow effect.

Inherited From
Gauge
Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isAnimated

Gets or sets a value that indicates whether the gauge animates value changes.

Inherited From
`Gauge`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can edit the value using the mouse and keyboard.

Inherited From
`Gauge`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

max

Gets or sets the maximum value that can be displayed on the gauge.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Gauge`
Type
number

min

Gets or sets the minimum value that can be displayed on the gauge.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Gauge`
Type
number

origin

Gets or sets the starting point used for painting the range.

By default, this property is set to null, which causes the value range to start at the gauge's minimum value, or zero if the minimum is less than zero.

Inherited From
`Gauge`
Type
number

pointer

Gets or sets the **`Range`** used to represent the gauge's current value.

Inherited From
`Gauge`
Type
`Range`

ranges

Gets the collection of ranges in this gauge.

Inherited From
`Gauge`
Type
`ObservableArray`

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showRanges

Gets or sets a value that indicates whether the gauge displays the ranges contained in the **ranges** property.

If this property is set to false, the ranges contained in the **ranges** property are not displayed in the gauge. Instead, they are used to interpolate the color of the **pointer** range while animating value changes.

Inherited From
Gauge
Type
boolean

showText

Gets or sets the **ShowText** values to display as text in the gauge.

Inherited From
Gauge
Type
ShowText

showTicks

Gets or sets a property that determines whether the gauge should display tickmarks at each **step** value.

The tickmarks can be formatted in CSS using the **wj-gauge** and **wj-ticks** class names. For example:

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

Inherited From
Gauge
Type
boolean

step

Gets or sets the amount to add to or subtract from the **value** property when the user presses the arrow keys or moves the mouse wheel.

Inherited From
Gauge
Type
number

thickness

Gets or sets the thickness of the gauge, on a scale between zero and one.

Setting the thickness to one causes the gauge to fill as much of the control area as possible. Smaller values create thinner gauges.

Inherited From
Gauge
Type
number

thumbSize

Gets or sets the size of the element that shows the gauge's current value, in pixels.

Inherited From
Gauge
Type
number

tickSpacing

Gets or sets the spacing between tickmarks.

Set the **showTicks** property to true if you want the gauge to show tickmarks along its face. By default, the interval between tickmarks is defined by the **step** property.

Use the **tickSpacing** property to override the default and use a spacing that is different from the **step** value. Set the **tickSpacing** property to null to revert to the default behavior.

Inherited From
Gauge
Type
number

value

Gets or sets the value displayed on the gauge.

Inherited From
Gauge
Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **Control**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **dispose** and **removeEventListener** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
Control
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): number
```

Gets a number that corresponds to the value of the gauge at a given point.

For example:

```
// hit test a point when the user clicks on the gauge
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates, or a MouseEvent object, or the x coordinate of the point.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
Gauge
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Gauge
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Indicates whether to update the control layout as well as the content.

Inherited From
Gauge
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes.

Inherited From
`Gauge`
Arguments
`EventArgs`

RadialGauge Class

File
`wijmo.gauge.js`
Module
wijmo.gauge
Base Class
Gauge
Derived Classes
WjRadialGauge

The **RadialGauge** displays a circular scale with an indicator that represents a single value and optional ranges to represent reference values.

If you set the gauge's **isReadOnly** property to false, then users will be able to edit the value by clicking on the gauge.

Example

Show me (http://jsfiddle.net/Wijmo5/kqkm8zt0)

Constructor

- constructor

Properties

- autoScale
- controlTemplate
- face
- format
- getText
- hasShadow
- hostElement
- isAnimated
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- max
- min
- origin
- pointer
- ranges
- rightToLeft
- showRanges
- showText
- showTicks
- startAngle
- step
- sweepAngle
- thickness
- thumbSize
- tickSpacing
- value

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): RadialGauge
```

Initializes a new instance of the **RadialGauge** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
RadialGauge

Properties

autoScale

Gets or sets a value that indicates whether the gauge automatically scales to fill the host element.

Type
boolean

STATIC controlTemplate

Gets or sets the template used to instantiate **Gauge** controls.

Inherited From
Gauge
Type
any

face

Gets or sets the **Range** used to represent the gauge's overall geometry and appearance.

Inherited From
Gauge
Type
Range

format

Gets or sets the format string used to display gauge values as text.

Inherited From
Gauge
Type
string

getText

Gets or sets a callback that returns customized strings used to display gauge values.

Use this property if you want to customize the strings shown on the gauge in cases where the **`format`** property is not enough.

If provided, the callback should be a function as that takes as parameters the gauge, the part name, the value, and the formatted value. The callback should return the string to be displayed on the gauge.

For example:

```
// callback to convert values into strings
gauge.getText = function (gauge, part, value, text) {
  switch (part) {
    case 'value':
      if (value <= 10) return 'Empty!';
      if (value <= 25) return 'Low...';
      if (value <= 95) return 'Good';
      return 'Full';
    case 'min':
      return 'EMPTY';
    case 'max':
      return 'FULL';
  }
  return text;
}
```

Inherited From
Gauge
Type
Function

hasShadow

Gets or sets a value that indicates whether the gauge displays a shadow effect.

Inherited From
Gauge
Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isAnimated

Gets or sets a value that indicates whether the gauge animates value changes.

Inherited From
`Gauge`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can edit the value using the mouse and keyboard.

Inherited From
`Gauge`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

max

Gets or sets the maximum value that can be displayed on the gauge.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Gauge`
Type
number

min

Gets or sets the minimum value that can be displayed on the gauge.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Gauge`
Type
number

origin

Gets or sets the starting point used for painting the range.

By default, this property is set to null, which causes the value range to start at the gauge's minimum value, or zero if the minimum is less than zero.

Inherited From
`Gauge`
Type
number

pointer

Gets or sets the **`Range`** used to represent the gauge's current value.

Inherited From
`Gauge`
Type
`Range`

ranges

Gets the collection of ranges in this gauge.

Inherited From
`Gauge`
Type
`ObservableArray`

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showRanges

Gets or sets a value that indicates whether the gauge displays the ranges contained in the **ranges** property.

If this property is set to false, the ranges contained in the **ranges** property are not displayed in the gauge. Instead, they are used to interpolate the color of the **pointer** range while animating value changes.

Inherited From
Gauge
Type
boolean

showText

Gets or sets the **ShowText** values to display as text in the gauge.

Inherited From
Gauge
Type
ShowText

showTicks

Gets or sets a property that determines whether the gauge should display tickmarks at each **step** value.

The tickmarks can be formatted in CSS using the **wj-gauge** and **wj-ticks** class names. For example:

```
.wj-gauge .wj-ticks {
    stroke-width: 2px;
    stroke: white;
}
```

Inherited From
Gauge
Type
boolean

startAngle

Gets or sets the starting angle for the gauge, in degrees.

Angles are measured in degrees, clockwise, starting from the 9 o'clock position.

Type
number

step

Gets or sets the amount to add to or subtract from the **`value`** property when the user presses the arrow keys or moves the mouse wheel.

Inherited From
Gauge
Type
number

sweepAngle

Gets or sets the sweeping angle for the gauge, in degrees.

Angles are measured in degrees, clockwise, starting from the 9 o'clock position.

Type
number

thickness

Gets or sets the thickness of the gauge, on a scale between zero and one.

Setting the thickness to one causes the gauge to fill as much of the control area as possible. Smaller values create thinner gauges.

Inherited From
Gauge
Type
number

thumbSize

Gets or sets the size of the element that shows the gauge's current value, in pixels.

Inherited From
Gauge
Type
number

tickSpacing

Gets or sets the spacing between tickmarks.

Set the **`showTicks`** property to true if you want the gauge to show tickmarks along its face. By default, the interval between tickmarks is defined by the **`step`** property.

Use the **`tickSpacing`** property to override the default and use a spacing that is different from the **`step`** value. Set the **`tickSpacing`** property to null to revert to the default behavior.

Inherited From
Gauge
Type
number

value

Gets or sets the value displayed on the gauge.

Inherited From
Gauge
Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): number
```

Gets a number that corresponds to the value of the gauge at a given point.

For example:

```
// hit test a point when the user clicks on the gauge
gauge.hostElement.addEventListener('click', function (e) {
  var ht = gauge.hitTest(e.pageX, e.pageY);
  if (ht != null) {
    console.log('you clicked the gauge at value ' + ht.toString());
  }
});
```

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates, or a MouseEvent object, or the x coordinate of the point.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
Gauge
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Gauge
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Indicates whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes.

Inherited From
`Gauge`
Arguments
`EventArgs`

Range Class

File
`wijmo.gauge.js`
Module
wijmo.gauge
Derived Classes
WjRange

Defines ranges to be used with **Gauge** controls.

Range objects have **min** and **max** properties that define the range's domain, as well as **color** and **thickness** properties that define the range's appearance.

Every **Gauge** control has at least two ranges: the 'face' defines the minimum and maximum values for the gauge, and the 'pointer' displays the gauge's current value.

In addition to the built-in ranges, gauges may have additional ranges used to display regions of significance (for example, low, medium, and high values).

Constructor

- constructor

Properties

- color
- max
- min
- name
- thickness

Methods

- onPropertyChanged

Events

- propertyChanged

Constructor

constructor

```
constructor(name?: string): Range
```

Initializes a new instance of the **Range** class.

Parameters

- **name**: **string** OPTIONAL
  The name of the range.

Returns
Range

Properties

color

Gets or sets the color used to display this range.

Type
string

max

Gets or sets the maximum value for this range.

Type
number

min

Gets or sets the minimum value for this range.

Type
number

name

Gets or sets the name of this **Range**.

Type
string

thickness

Gets or sets the thickness of this range as a percentage of the parent gauge's thickness.

Type
number

Methods

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

Raises the **propertyChanged** event.

Parameters

- **e**: **PropertyChangedEventArgs**
  `PropertyChangedEventArgs` that contains the property name, old, and new values.

Returns
void

Events

propertyChanged

Occurs when the value of a property in this **Range** changes.

Arguments
`PropertyChangedEventArgs`

GaugeDirection Enum

File
`wijmo.gauge.js`
Module
wijmo.gauge

Represents the direction in which the pointer of a **LinearGauge** increases.

Members

Name	Value	Description
Right	0	Gauge value increases from left to right.
Left	1	Gauge value increases from right to left.
Up	2	Gauge value increases from bottom to top.
Down	3	Gauge value increases from top to bottom.

ShowText Enum

File
`wijmo.gauge.js`
Module
wijmo.gauge

Specifies which values to display as text.

Members

Name	Value	Description
None	0	Do not show any text in the gauge.
Value	1	Show the gauge's **`value`** as text.
MinMax	2	Show the gauge's **`min`** and **`max`** values as text.
All	3	Show the gauge's **`value`**, **`min`**, and **`max`** as text.

wijmo.odata Module

File
`wijmo.odata.js`
Module
wijmo.odata

Provides classes that support the OData protocol, including the **ODataCollectionView** class.

OData is a standardized protocol for creating and consuming data APIs. OData builds on core protocols like HTTP and commonly accepted methodologies like REST. The result is a uniform way to expose full-featured data APIs. (http://www.odata.org/)

Classes

- ODataCollectionView
- ODataVirtualCollectionView

ODataCollectionView Class

File
`wijmo.odata.js`
Module
wijmo.odata
Base Class
CollectionView
Derived Classes
ODataVirtualCollectionView

Extends the **CollectionView** class to support loading and saving data to and from OData sources.

You can use the **ODataCollectionView** class to load data from OData services and use it as a data source for any Wijmo controls.

In addition to full CRUD support you get all the **CollectionView** features including sorting, filtering, paging, and grouping. The sorting, filtering, and paging functions may be performed on the server or on the client.

The code below shows how you can instantiate an **ODataCollectionView** that selects some fields from the data source and provides sorting on the client. Notice how the 'options' parameter is used to pass in initialization data, which is the same approach used when initializing controls:

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var categories = new wijmo.odata.ODataCollectionView(url, 'Categories', {
  fields: ['CategoryID', 'CategoryName', 'Description'],
  sortOnServer: false
});
```

Constructor

- constructor

Properties

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- dataTypes
- fields
- filter
- filterDefinition
- filterOnServer
- getError
- groupDescriptions
- groups
- inferDataTypes
- isAddingNew
- isEditingItem
- isEmpty
- isLoading
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- keys
- newItemCreator
- oDataVersion
- pageCount
- pageIndex
- pageOnServer
- pageSize
- requestHeaders
- sortComparer
- sortConverter
- sortDescriptions
- sortOnServer
- sourceCollection
- tableName
- totalItemCount
- trackChanges
- useStableSort

Methods

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- load
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onError
- onLoaded
- onLoading
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh

- remove
- removeAt
- updateFilterDefinition

Events

- collectionChanged
- currentChanged
- currentChanging
- error
- loaded
- loading
- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

Constructor

constructor

```
constructor(url: string, tableName: string, options?: any): ODataCollectionView
```

Initializes a new instance of the **`ODataCollectionView`** class.

Parameters

- **url**: **string**
  Url of the OData service (for example 'http://services.odata.org/Northwind/Northwind.svc').
- **tableName**: **string**
  Name of the table (entity) to retrieve from the service. If not provided, a list of the tables (entities) available is retrieved.
- **options**: **any** OPTIONAL
  JavaScript object containing initialization data (property values and event handlers) for the **`ODataCollectionView`**.

Returns
ODataCollectionView

Properties

- **canAddNew**

Gets a value that indicates whether a new item can be added to the collection.

Inherited From
`CollectionView`
Type
boolean

canCancelEdit

Gets a value that indicates whether the collection view can discard pending changes and restore the original values of an edited object.

Inherited From
`CollectionView`
Type
boolean

canChangePage

Gets a value that indicates whether the **`pageIndex`** value can change.

Inherited From
`CollectionView`
Type
boolean

canFilter

Gets a value that indicates whether this view supports filtering via the **`filter`** property.

Inherited From
`CollectionView`
Type
boolean

canGroup

Gets a value that indicates whether this view supports grouping via the **`groupDescriptions`** property.

Inherited From
`CollectionView`
Type
boolean

canRemove

Gets a value that indicates whether items can be removed from the collection.

Inherited From
`CollectionView`
Type
boolean

canSort

Gets a value that indicates whether this view supports sorting via the **`sortDescriptions`** property.

Inherited From
`CollectionView`
Type
boolean

currentAddItem

Gets the item that is being added during the current add transaction.

Inherited From
`CollectionView`
Type
any

currentEditItem

Gets the item that is being edited during the current edit transaction.

Inherited From
`CollectionView`
Type
any

currentItem

Gets or sets the current item in the view.

Inherited From
`CollectionView`
Type
any

currentPosition

Gets the ordinal position of the current item in the view.

Inherited From
`CollectionView`
Type
number

dataTypes

Gets or sets a JavaScript object to be used as a map for coercing data types when loading the data.

The object keys represent the field names and the values are **`DataType`** values that indicate how the data should be coerced.

For example, the code below creates an **`ODataCollectionView`** and specifies that 'Freight' values, which are stored as strings in the database, should be converted into numbers; and that three date fields should be converted into dates:

```
var orders = new wijmo.data.ODataCollectionView(url, 'Orders', {
  dataTypes: {
    Freight: wijmo.DataType.Number
    OrderDate: wijmo.DataType.Date,
    RequiredDate: wijmo.DataType.Date,
    ShippedDate: wijmo.DataType.Date,
  }
});
```

This property is useful when the database contains data stored in formats that do not conform to common usage.

In most cases you don't have to provide information about the data types, because the **`inferDataTypes`** property handles the conversion of Date values automatically.

If you do provide explicit type information, the **`inferDataTypes`** property is not applied. Because of this, any data type information that is provided should be complete, including all fields of type Date.

Type
any

fields

Gets or sets an array containing the names of the fields to retrieve from the data source.

If this property is set to null or to an empty array, all fields are retrieved.

For example, the code below creates an **ODataCollectionView** that gets only three fields from the 'Categories' table in the database:

```
var categories = new wijmo.data.ODataCollectionView(url, 'Categories', {
  fields: ['CategoryID', 'CategoryName', 'Description']
});
```

Type
string[]

filter

Gets or sets a callback used to determine if an item is suitable for inclusion in the view.

The callback function should return true if the item passed in as a parameter should be included in the view.

NOTE: If the filter function needs a scope (i.e. a meaningful 'this' value) remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
collectionView.filter = this._filter.bind(this);
```

Inherited From
CollectionView
Type
IPredicate

filterDefinition

Gets or sets a string containing an OData filter specification to be used for filtering the data on the server.

The filter definition syntax is described in the **OData documentation**.

For example, the code below causes the server to return records where the 'CompanyName' field starts with 'A' and ends with 'S':

```
view.filterDefinition = "startswith(CompanyName, 'A') and endswith(CompanyName, 'B')";
```

Filter definitions can be generated automatically. For example, the **FlexGridFilter** component detects whether its data source is an **ODataCollectionView** and automatically updates both the **filter** and **filterDefinition** properties.

Note that the **filterDefinition** property is applied even if the **filterOnServer** property is set to false. This allows you to apply server and client filters to the same collection, which can be useful in many scenarios.

For example, the code below uses the **filterDefinition** property to filter on the server and the **filter** property to further filter on the client. The collection will show items with names that start with 'C' and have unit prices greater than 20:

```
var url = 'http://services.odata.org/V4/Northwind/Northwind.svc/';
var data = new wijmo.odata.ODataCollectionView(url, 'Products', {
  oDataVersion: 4,
  filterDefinition: 'startswith(ProductName, \'C\')', // server filter
  filterOnServer: false, // client filter
  filter: function(product) {
    return product.UnitPrice > 20;
  },
});
```

Type
string

filterOnServer

Gets or sets a value that determines whether filtering should be performed on the server or on the client.

Use the **filter** property to perform filtering on the client, and use the **filterDefinition** property to perform filtering on the server.

In some cases it may be desirable to apply independent filters on the client **and** on the server.

You can achieve this by setting (1) the **filterOnServer** property to false and the **filter** property to a filter function (to enable client-side filtering) and (2) the **filterDefinition** property to a filter string (to enable server-side filtering).

Type
boolean

getError

Gets or sets a callback that determines whether a specific property of an item contains validation errors.

If provided, the callback should take two parameters containing the item and the property to validate, and should return a string describing the error (or null if there are no errors).

For example:

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? 'Cannot be negative!'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'No active items allowed in the US or UK!'
                    : null;
        }
        return null;
    }
});
```

Inherited From
`CollectionView`
Type
Function

groupDescriptions

Gets a collection of **`GroupDescription`** objects that describe how the items in the collection are grouped in the view.

Inherited From
`CollectionView`
Type
`ObservableArray`

groups

Gets an array of **`CollectionViewGroup`** objects that represents the top-level groups.

Inherited From
`CollectionView`
Type
`CollectionViewGroup`[]

inferDataTypes

Gets or sets a value that determines whether fields that contain strings that look like standard date representations should be converted to dates automatically.

This property is set to true by default, because the **`ODataCollectionView`** class uses JSON and that format does not support Date objects.

This property has no effect if specific type information is provided using the **`dataTypes`** property.

Type
boolean

isAddingNew

Gets a value that indicates whether an add transaction is in progress.

Inherited From
`CollectionView`
Type
boolean

isEditingItem

Gets a value that indicates whether an edit transaction is in progress.

Inherited From
`CollectionView`
Type
boolean

isEmpty

Gets a value that indicates whether this view contains no items.

Inherited From
`CollectionView`
Type
boolean

isLoading

Gets a value that indicates the `ODataCollectionView` is currently loading data.

This property can be used to provide progress indicators.

Type
boolean

isPageChanging

Gets a value that indicates whether the page index is changing.

Inherited From
`CollectionView`
Type
boolean

isUpdating

Gets a value that indicates whether notifications are currently suspended (see `beginUpdate` and `endUpdate`).

Inherited From
`CollectionView`
Type

itemCount

Gets the total number of items in the view taking paging into account.

Inherited From
`CollectionView`
Type
number

items

Gets items in the view.

Inherited From
`CollectionView`
Type
any[]

itemsAdded

Gets an **`ObservableArray`** containing the records that were added to the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

itemsEdited

Gets an **`ObservableArray`** containing the records that were edited in the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

itemsRemoved

Gets an **`ObservableArray`** containing the records that were removed from the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

keys

Gets or sets an array containing the names of the key fields.

Key fields are required for update operations (add/remove/delete).

Type
string[]

newItemCreator

Gets or sets a function that creates new items for the collection.

If the creator function is not supplied, the **`CollectionView`** will try to create an uninitialized item of the appropriate type.

If the creator function is supplied, it should be a function that takes no parameters and returns an initialized object of the proper type for the collection.

Inherited From
`CollectionView`
Type
Function

oDataVersion

Gets or sets the OData version used by the server.

There are currently four versions of OData services, 1.0 through 4.0. Version 4.0 is used by the latest services, but there are many legacy services still in operation.

If you know what version of OData your service implements, set the **oDataVersion** property to the appropriate value (1 through 4) when creating the **ODataCollectionView** (see example below).

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var categories = new wijmo.odata.ODataCollectionView(url, 'Categories', {
  oDataVersion: 1.0, // legacy OData source
  fields: ['CategoryID', 'CategoryName', 'Description'],
  sortOnServer: false
});
```

If you do not know what version of OData your service implements (perhaps you are writing an OData explorer application), then do not specify the version. In this case, the **ODataCollectionView** will get this information from the server. This operation requires an extra request, but only once per service URL, so the overhead is small.

Type
number

pageCount

Gets the total number of pages.

Type
number

pageIndex

Gets the zero-based index of the current page.

Inherited From
CollectionView
Type
number

pageOnServer

Gets or sets a value that determines whether paging should be performed on the server or on the client.

Use the **pageSize** property to enable paging.

Type
boolean

pageSize

Gets or sets the number of items to display on a page.

Type
number

requestHeaders

Gets or sets an object containing request headers to be used when sending or requesting data.

The most typical use for this property is in scenarios where authentication is required. For example:

```
var categories = new wijmo.odata.ODataCollectionView(serviceUrl, 'Categories', {
  fields: ['Category_ID', 'Category_Name'],
  requestHeaders: { Authorization: db.token }
});
```

Type
any

sortComparer

Gets or sets a function used to compare values when sorting.

If provided, the sort comparer function should take as parameters two values of any type, and should return -1, 0, or +1 to indicate whether the first value is smaller than, equal to, or greater than the second. If the sort comparer returns null, the standard built-in comparer is used.

This **`sortComparer`** property allows you to use custom comparison algorithms that in some cases result in sorting sequences that are more consistent with user's expectations than plain string comparisons.

For example, see **Dave Koele's Alphanum algorithm**. It breaks up strings into chunks composed of strings or numbers, then sorts number chunks in value order and string chunks in ASCII order. Dave calls the result a "natural sorting order".

The example below shows a typical use for the **`sortComparer`** property:

```
// create a CollectionView with a custom sort comparer
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // custom comparer used for strings
      : null; // use default comparer used for everything else
  }
});
```

Inherited From
`CollectionView`
Type
Function

sortConverter

Gets or sets a function used to convert values when sorting.

If provided, the function should take as parameters a **SortDescription**, a data item, and a value to convert, and should return the converted value.

This property provides a way to customize sorting. For example, the **FlexGrid** control uses it to sort mapped columns by display value instead of by raw value.

For example, the code below causes a **CollectionView** to sort the 'country' property, which contains country code integers, using the corresponding country names:

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // convert country id into name
  }
  return value;
}
```

Inherited From
CollectionView
Type
Function

sortDescriptions

Gets a collection of **SortDescription** objects that describe how the items in the collection are sorted in the view.

Inherited From
CollectionView
Type
ObservableArray

sortOnServer

Gets or sets a value that determines whether sort operations should be performed on the server or on the client.

Use the **sortDescriptions** property to specify how the data should be sorted.

Type
boolean

sourceCollection

Gets or sets the underlying (unfiltered and unsorted) collection.

Inherited From
`CollectionView`
Type
any

tableName

Gets the name of the table (entity) that this collection is bound to.

Type
string

totalItemCount

Gets the total number of items in the view before paging is applied.

Type
number

trackChanges

Gets or sets a value that determines whether the control should track changes to the data.

If **`trackChanges`** is set to true, the **`CollectionView`** keeps track of changes to the data and exposes them through the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

Tracking changes is useful in situations where you need to update the server after the user has confirmed that the modifications are valid.

After committing or cancelling changes, use the **`clearChanges`** method to clear the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

The **`CollectionView`** only tracks changes made when the proper **`CollectionView`** methods are used (**`editItem`**/**`commitEdit`**, **`addNew`**/**`commitNew`**, and **`remove`**). Changes made directly to the data are not tracked.

Inherited From
`CollectionView`
Type
boolean

useStableSort

Gets or sets whether to use a stable sort algorithm.

Stable sorting algorithms maintain the relative order of records with equal keys. For example, consider a collection of objects with an "Amount" field. If you sort the collection by "Amount", a stable sort will keep the original order of records with the same Amount value.

This property is false by default, which causes the **`CollectionView`** to use JavaScript's built-in sort method, which is very fast but not stable. Setting the **`useStableSort`** property to true increases sort times by 30% to 50%, which can be significant for large collections.

Inherited From
`CollectionView`
Type
boolean

Methods

addNew

```
addNew(): any
```

Creates a new item and adds it to the collection.

This method takes no parameters. It creates a new item, adds it to the collection, and defers refresh operations until the new item is committed using the **commitNew** method or canceled using the **cancelNew** method.

The code below shows how the **addNew** method is typically used:

```
// create the new item, add it to the collection
var newItem = view.addNew();

// initialize the new item
newItem.id = getFreshId();
newItem.name = 'New Customer';

// commit the new item so the view can be refreshed
view.commitNew();
```

You can also add new items by pushing them into the **sourceCollection** and then calling the **refresh** method. The main advantage of **addNew** is in user-interactive scenarios (like adding new items in a data grid), because it gives users the ability to cancel the add operation. It also prevents the new item from being sorted or filtered out of view until the add operation is committed.

Inherited From
CollectionView
Returns
any

beginUpdate

```
beginUpdate(): void
```

Suspend refreshes until the next call to **endUpdate**.

Inherited From
CollectionView
Returns
void

cancelEdit

```
cancelEdit(): void
```

Ends the current edit transaction and, if possible, restores the original value to the item.

Inherited From
`CollectionView`
Returns
void

cancelNew

```
cancelNew(): void
```

Ends the current add transaction and discards the pending new item.

Inherited From
`CollectionView`
Returns
void

clearChanges

```
clearChanges(): void
```

Clears all changes by removing all items in the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

Call this method after committing changes to the server or after refreshing the data from the server.

Inherited From
`CollectionView`
Returns
void

commitEdit

```
commitEdit(): void
```

Override **`commitEdit`** to modify the item in the database.

Returns
void

commitNew

```
commitNew(): void
```

Override **`commitNew`** to add the new item to the database.

Returns
void

contains

```
contains(item: any): boolean
```

Returns a value indicating whether a given item belongs to this view.

Parameters

- **item**: **any**
  Item to seek.

Inherited From
`CollectionView`
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The collection will not be refreshed until the function finishes. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
`CollectionView`
Returns
void

editItem

```
editItem(item: any): void
```

Begins an edit transaction of the specified item.

Parameters

- **item**: **any**
  Item to be edited.

Inherited From
`CollectionView`
Returns
void

endUpdate

```
endUpdate(): void
```

Resume refreshes suspended by a call to **beginUpdate**.

Inherited From
CollectionView
Returns
void

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

Calculates an aggregate value for the items in this collection.

Parameters

- **aggType**: **Aggregate**
  Type of aggregate to calculate.
- **binding**: **string**
  Property to aggregate on.
- **currentPage**: **boolean** OPTIONAL
  Whether to include only items on the current page.

Inherited From
CollectionView
Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
CollectionView
Returns
boolean

load

```
load(): void
```

Loads or re-loads the data from the OData source.

Returns
void

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

Sets the specified item to be the current item in the view.

Parameters

- **item**: **any**
  Item that will become current.

Inherited From
CollectionView
Returns
boolean

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

Sets the first item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

Sets the last item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

Sets the item after the current item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

Sets the item at the specified index in the view as the current item.

Parameters

- **index**: **number**
  Index of the item that will become current.

Inherited From
CollectionView
Returns
boolean

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

Sets the item before the current item in the view as the current item.

Inherited From
CollectionView
Returns
boolean

moveToFirstPage

```
moveToFirstPage(): boolean
```

Sets the first page as the current page.

Inherited From
CollectionView
Returns
boolean

moveToLastPage

```
moveToLastPage(): boolean
```

Sets the last page as the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToNextPage

```
moveToNextPage(): boolean
```

Moves to the page after the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToPage

```
moveToPage(index: number): boolean
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index of the page to move to.

Inherited From
`CollectionView`
Returns
boolean

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

Moves to the page before the current page.

Inherited From
CollectionView
Returns
boolean

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
CollectionView
Returns
void

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

Raises the **currentChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
CollectionView
Returns
void

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

Raises the **currentChanging** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
CollectionView
Returns
boolean

onError

```
onError(e: RequestErrorEventArgs): boolean
```

Raises the **error** event.

By default, errors throw exceptions and trigger a data refresh. If you want to prevent this behavior, set the **cancel** parameter to true in the event handler.

Parameters

- **e**: **RequestErrorEventArgs**
  RequestErrorEventArgs that contains information about the error.

Returns
boolean

onLoaded

```
onLoaded(e?: EventArgs): void
```

Raises the **loaded** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoading

```
onLoading(e?: EventArgs): void
```

Raises the **loading** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

Raises the **pageChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`CollectionView`
Returns
void

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

Raises the **pageChanging** event.

Parameters

- **e**: **PageChangingEventArgs**
  `PageChangingEventArgs` that contains the event data.

Returns
boolean

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

Raises the **sourceCollectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`CollectionView`
Returns
void

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

Raises the **`sourceCollectionChanging`** event.

Parameters

- **e**: **CancelEventArgs**
  `CancelEventArgs` that contains the event data.

Inherited From
`CollectionView`
Returns
boolean

refresh

```
refresh(): void
```

Re-creates the view using the current sort, filter, and group parameters.

Inherited From
`CollectionView`
Returns
void

remove

```
remove(item: any): void
```

Override **`remove`** to remove the item from the database.

Parameters

- **item**: **any**
  Item to be removed from the database.

Returns
void

removeAt

```
removeAt(index: number): void
```

Removes the item at the specified index from the collection.

Parameters

- **index**: **number**
  Index of the item to be removed from the collection. The index is relative to the view, not to the source collection.

Inherited From
`CollectionView`
Returns
void

updateFilterDefinition

```
updateFilterDefinition(filterProvider: any): void
```

Updates the filter definition based on a known filter provider such as the **`FlexGridFilter`**.

Parameters

- **filterProvider**: **any**
  Known filter provider, typically an instance of a **`FlexGridFilter`**.

Returns
void

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`CollectionView`
Arguments
`NotifyCollectionChangedEventArgs`

currentChanged

Occurs after the current item changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

currentChanging

Occurs before the current item changes.

Inherited From
`CollectionView`
Arguments
`CancelEventArgs`

error

Occurs when there is an error reading or writing data.

Arguments
`RequestErrorEventArgs`

loaded

Occurs when the `ODataCollectionView` finishes loading data.

Arguments
`EventArgs`

loading

Occurs when the `ODataCollectionView` starts loading data.

Arguments
`EventArgs`

pageChanged

Occurs after the page index changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

pageChanging

Occurs before the page index changes.

Inherited From
`CollectionView`
Arguments
`PageChangingEventArgs`

sourceCollectionChanged

Occurs after the value of the **`sourceCollection`** property changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

sourceCollectionChanging

Occurs before the value of the **`sourceCollection`** property changes.

Inherited From
`CollectionView`
Arguments
`CancelEventArgs`

ODataVirtualCollectionView Class

File
`wijmo.odata.js`
Module
wijmo.odata
Base Class
ODataCollectionView

Extends the **ODataCollectionView** class to support loading data on demand, using the **setWindow** method.

The example below shows how you can declare an **ODataCollectionView** and synchronize it with a **FlexGrid** control to load the data that is within the grid's viewport:

```
// declare virtual collection view
var vcv = new wijmo.odata.ODataVirtualCollectionView(url, 'Order_Details_Extendeds', {
  oDataVersion: 4
});

// use virtual collection as grid data source
flex.itemsSource = vcv;

// update data window when the grid scrolls
flex.scrollPositionChanged.addHandler(function () {
  var rng = flex.viewRange;
  vcv.setWindow(rng.row, rng.row2);
});
```

The **ODataVirtualCollectionView** class implements a 'data window' so only data that is actually being displayed is loaded from the server. Items that are not being displayed are added to the collection as null values until a call to the **setWindow** method causes them those items to be loaded.

This 'on-demand' method of loading data has advantages when dealing with large data sets, because it prevents the application from loading data until it is required. But it does impose some limitation: sorting and filtering must be done on the server; grouping and paging are not supported.

Constructor

- constructor

Properties

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- dataTypes
- fields
- filter
- filterDefinition
- filterOnServer
- getError
- groupDescriptions
- groups
- inferDataTypes
- isAddingNew
- isEditingItem
- isEmpty
- isLoading
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- keys
- newItemCreator
- oDataVersion
- pageCount
- pageIndex
- pageOnServer
- pageSize
- requestHeaders
- sortComparer
- sortConverter
- sortDescriptions
- sortOnServer
- sourceCollection
- tableName
- totalItemCount
- trackChanges
- useStableSort

Methods

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- load
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onError
- onLoaded
- onLoading
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh

- remove
- removeAt
- setWindow
- updateFilterDefinition

Events

- collectionChanged
- currentChanged
- currentChanging
- error
- loaded
- loading
- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

Constructor

constructor

```
constructor(url: string, tableName: string, options?: any): ODataVirtualCollectionView
```

Initializes a new instance of the **`ODataVirtualCollectionView`** class.

Parameters

- **url**: **string**
  Url of the OData service (for example 'http://services.odata.org/Northwind/Northwind.svc').
- **tableName**: **string**
  Name of the table (entity) to retrieve from the service. If not provided, a list of the tables (entities) available is retrieved.
- **options**: **any** OPTIONAL
  JavaScript object containing initialization data (property values and event handlers) for the **`ODataVirtualCollectionView`**.

Returns
ODataVirtualCollectionView

Properties

canAddNew

Gets a value that indicates whether a new item can be added to the collection.

Inherited From
`CollectionView`
Type
boolean

canCancelEdit

Gets a value that indicates whether the collection view can discard pending changes and restore the original values of an edited object.

Inherited From
`CollectionView`
Type
boolean

canChangePage

Gets a value that indicates whether the **`pageIndex`** value can change.

Inherited From
`CollectionView`
Type
boolean

canFilter

Gets a value that indicates whether this view supports filtering via the **`filter`** property.

Inherited From
`CollectionView`
Type
boolean

canGroup

`ODataVirtualCollectionView` requires **`canGroup`** to be set to false.

Type
boolean

canRemove

Gets a value that indicates whether items can be removed from the collection.

Inherited From
`CollectionView`
Type
boolean

canSort

Gets a value that indicates whether this view supports sorting via the **`sortDescriptions`** property.

Inherited From
`CollectionView`
Type
boolean

currentAddItem

Gets the item that is being added during the current add transaction.

Inherited From
`CollectionView`
Type
any

currentEditItem

Gets the item that is being edited during the current edit transaction.

Inherited From
`CollectionView`
Type
any

currentItem

Gets or sets the current item in the view.

Inherited From
`CollectionView`
Type
any

currentPosition

Gets the ordinal position of the current item in the view.

Inherited From
`CollectionView`
Type
number

dataTypes

Gets or sets a JavaScript object to be used as a map for coercing data types when loading the data.

The object keys represent the field names and the values are **DataType** values that indicate how the data should be coerced.

For example, the code below creates an **ODataCollectionView** and specifies that 'Freight' values, which are stored as strings in the database, should be converted into numbers; and that three date fields should be converted into dates:

```
var orders = new wijmo.data.ODataCollectionView(url, 'Orders', {
  dataTypes: {
    Freight: wijmo.DataType.Number
    OrderDate: wijmo.DataType.Date,
    RequiredDate: wijmo.DataType.Date,
    ShippedDate: wijmo.DataType.Date,
  }
});
```

This property is useful when the database contains data stored in formats that do not conform to common usage.

In most cases you don't have to provide information about the data types, because the **inferDataTypes** property handles the conversion of Date values automatically.

If you do provide explicit type information, the **inferDataTypes** property is not applied. Because of this, any data type information that is provided should be complete, including all fields of type Date.

Inherited From
ODataCollectionView
Type
any

fields

Gets or sets an array containing the names of the fields to retrieve from the data source.

If this property is set to null or to an empty array, all fields are retrieved.

For example, the code below creates an **ODataCollectionView** that gets only three fields from the 'Categories' table in the database:

```
var categories = new wijmo.data.ODataCollectionView(url, 'Categories', {
  fields: ['CategoryID', 'CategoryName', 'Description']
});
```

Inherited From
ODataCollectionView
Type
string[]

filter

Gets or sets a callback used to determine if an item is suitable for inclusion in the view.

The callback function should return true if the item passed in as a parameter should be included in the view.

NOTE: If the filter function needs a scope (i.e. a meaningful 'this' value) remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
collectionView.filter = this._filter.bind(this);
```

Inherited From
`CollectionView`
Type
`IPredicate`

filterDefinition

Gets or sets a string containing an OData filter specification to be used for filtering the data on the server.

The filter definition syntax is described in the **OData documentation**.

For example, the code below causes the server to return records where the 'CompanyName' field starts with 'A' and ends with 'S':

```
view.filterDefinition = "startswith(CompanyName, 'A') and endswith(CompanyName, 'B')";
```

Filter definitions can be generated automatically. For example, the **`FlexGridFilter`** component detects whether its data source is an **`ODataCollectionView`** and automatically updates both the **`filter`** and **`filterDefinition`** properties.

Note that the **`filterDefinition`** property is applied even if the **`filterOnServer`** property is set to false. This allows you to apply server and client filters to the same collection, which can be useful in many scenarios.

For example, the code below uses the **`filterDefinition`** property to filter on the server and the **`filter`** property to further filter on the client. The collection will show items with names that start with 'C' and have unit prices greater than 20:

```
var url = 'http://services.odata.org/V4/Northwind/Northwind.svc/';
var data = new wijmo.odata.ODataCollectionView(url, 'Products', {
  oDataVersion: 4,
  filterDefinition: 'startswith(ProductName, \'C\')', // server filter
  filterOnServer: false, // client filter
  filter: function(product) {
    return product.UnitPrice > 20;
  },
});
```

Inherited From
`ODataCollectionView`
Type
string

filterOnServer

`ODataVirtualCollectionView` requires **`filterOnServer`** to be set to true.

Type
boolean

getError

Gets or sets a callback that determines whether a specific property of an item contains validation errors.

If provided, the callback should take two parameters containing the item and the property to validate, and should return a string describing the error (or null if there are no errors).

For example:

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? 'Cannot be negative!'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'No active items allowed in the US or UK!'
                    : null;
        }
        return null;
    }
});
```

Inherited From
`CollectionView`
Type
Function

groupDescriptions

Gets a collection of **`GroupDescription`** objects that describe how the items in the collection are grouped in the view.

Inherited From
`CollectionView`
Type
`ObservableArray`

groups

Gets an array of **`CollectionViewGroup`** objects that represents the top-level groups.

Inherited From
`CollectionView`
Type
`CollectionViewGroup`[]

inferDataTypes

Gets or sets a value that determines whether fields that contain strings that look like standard date representations should be converted to dates automatically.

This property is set to true by default, because the **`ODataCollectionView`** class uses JSON and that format does not support Date objects.

This property has no effect if specific type information is provided using the **`dataTypes`** property.

Inherited From
`ODataCollectionView`
Type
boolean

isAddingNew

Gets a value that indicates whether an add transaction is in progress.

Inherited From
`CollectionView`
Type
boolean

isEditingItem

Gets a value that indicates whether an edit transaction is in progress.

Inherited From
`CollectionView`
Type
boolean

isEmpty

Gets a value that indicates whether this view contains no items.

Inherited From
`CollectionView`
Type
boolean

isLoading

Gets a value that indicates the `ODataCollectionView` is currently loading data.

This property can be used to provide progress indicators.

Inherited From
`ODataCollectionView`
Type
boolean

isPageChanging

Gets a value that indicates whether the page index is changing.

Inherited From
`CollectionView`
Type
boolean

isUpdating

Gets a value that indicates whether notifications are currently suspended (see `beginUpdate` and `endUpdate`).

Inherited From
`CollectionView`
Type

itemCount

Gets the total number of items in the view taking paging into account.

Inherited From
`CollectionView`
Type
number

items

Gets items in the view.

Inherited From
`CollectionView`
Type
any[]

itemsAdded

Gets an **`ObservableArray`** containing the records that were added to the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

itemsEdited

Gets an **`ObservableArray`** containing the records that were edited in the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

itemsRemoved

Gets an **ObservableArray** containing the records that were removed from the collection since **trackChanges** was enabled.

Inherited From
CollectionView
Type
ObservableArray

keys

Gets or sets an array containing the names of the key fields.

Key fields are required for update operations (add/remove/delete).

Inherited From
ODataCollectionView
Type
string[]

newItemCreator

Gets or sets a function that creates new items for the collection.

If the creator function is not supplied, the **CollectionView** will try to create an uninitialized item of the appropriate type.

If the creator function is supplied, it should be a function that takes no parameters and returns an initialized object of the proper type for the collection.

Inherited From
CollectionView
Type
Function

oDataVersion

Gets or sets the OData version used by the server.

There are currently four versions of OData services, 1.0 through 4.0. Version 4.0 is used by the latest services, but there are many legacy services still in operation.

If you know what version of OData your service implements, set the **oDataVersion** property to the appropriate value (1 through 4) when creating the **ODataCollectionView** (see example below).

```
var url = 'http://services.odata.org/Northwind/Northwind.svc';
var categories = new wijmo.odata.ODataCollectionView(url, 'Categories', {
  oDataVersion: 1.0, // legacy OData source
  fields: ['CategoryID', 'CategoryName', 'Description'],
  sortOnServer: false
});
```

If you do not know what version of OData your service implements (perhaps you are writing an OData explorer application), then do not specify the version. In this case, the **ODataCollectionView** will get this information from the server. This operation requires an extra request, but only once per service URL, so the overhead is small.

Inherited From
ODataCollectionView
Type
number

pageCount

Gets the total number of pages.

Inherited From
ODataCollectionView
Type
number

pageIndex

Gets the zero-based index of the current page.

Inherited From
CollectionView
Type
number

pageOnServer

ODataVirtualCollectionView requires **pageOnServer** to be set to true.

Type
boolean

pageSize

Gets or sets the number of items to display on a page.

Inherited From
ODataCollectionView
Type
number

requestHeaders

Gets or sets an object containing request headers to be used when sending or requesting data.

The most typical use for this property is in scenarios where authentication is required. For example:

```
var categories = new wijmo.odata.ODataCollectionView(serviceUrl, 'Categories', {
  fields: ['Category_ID', 'Category_Name'],
  requestHeaders: { Authorization: db.token }
});
```

Inherited From
ODataCollectionView
Type
any

sortComparer

Gets or sets a function used to compare values when sorting.

If provided, the sort comparer function should take as parameters two values of any type, and should return -1, 0, or +1 to indicate whether the first value is smaller than, equal to, or greater than the second. If the sort comparer returns null, the standard built-in comparer is used.

This **`sortComparer`** property allows you to use custom comparison algorithms that in some cases result in sorting sequences that are more consistent with user's expectations than plain string comparisons.

For example, see **Dave Koele's Alphanum algorithm**. It breaks up strings into chunks composed of strings or numbers, then sorts number chunks in value order and string chunks in ASCII order. Dave calls the result a "natural sorting order".

The example below shows a typical use for the **`sortComparer`** property:

```
// create a CollectionView with a custom sort comparer
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // custom comparer used for strings
      : null; // use default comparer used for everything else
  }
});
```

Inherited From
`CollectionView`
Type
Function

sortConverter

Gets or sets a function used to convert values when sorting.

If provided, the function should take as parameters a **SortDescription**, a data item, and a value to convert, and should return the converted value.

This property provides a way to customize sorting. For example, the **FlexGrid** control uses it to sort mapped columns by display value instead of by raw value.

For example, the code below causes a **CollectionView** to sort the 'country' property, which contains country code integers, using the corresponding country names:

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // convert country id into name
  }
  return value;
}
```

Inherited From
CollectionView
Type
Function

sortDescriptions

Gets a collection of **SortDescription** objects that describe how the items in the collection are sorted in the view.

Inherited From
CollectionView
Type
ObservableArray

sortOnServer

ODataVirtualCollectionView requires **sortOnServer** to be set to true.

Type
boolean

sourceCollection

Gets or sets the underlying (unfiltered and unsorted) collection.

Inherited From
`CollectionView`
Type
any

tableName

Gets the name of the table (entity) that this collection is bound to.

Inherited From
`ODataCollectionView`
Type
string

totalItemCount

Gets the total number of items in the view before paging is applied.

Inherited From
`ODataCollectionView`
Type
number

trackChanges

Gets or sets a value that determines whether the control should track changes to the data.

If **`trackChanges`** is set to true, the **`CollectionView`** keeps track of changes to the data and exposes them through the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

Tracking changes is useful in situations where you need to update the server after the user has confirmed that the modifications are valid.

After committing or cancelling changes, use the **`clearChanges`** method to clear the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

The **`CollectionView`** only tracks changes made when the proper **`CollectionView`** methods are used (**`editItem`**/**`commitEdit`**, **`addNew`**/**`commitNew`**, and **`remove`**). Changes made directly to the data are not tracked.

Inherited From
`CollectionView`
Type
boolean

useStableSort

Gets or sets whether to use a stable sort algorithm.

Stable sorting algorithms maintain the relative order of records with equal keys. For example, consider a collection of objects with an "Amount" field. If you sort the collection by "Amount", a stable sort will keep the original order of records with the same Amount value.

This property is false by default, which causes the **`CollectionView`** to use JavaScript's built-in sort method, which is very fast but not stable. Setting the **`useStableSort`** property to true increases sort times by 30% to 50%, which can be significant for large collections.

Inherited From
`CollectionView`
Type
boolean

Methods

addNew

```
addNew(): any
```

Creates a new item and adds it to the collection.

This method takes no parameters. It creates a new item, adds it to the collection, and defers refresh operations until the new item is committed using the **commitNew** method or canceled using the **cancelNew** method.

The code below shows how the **addNew** method is typically used:

```
// create the new item, add it to the collection
var newItem = view.addNew();

// initialize the new item
newItem.id = getFreshId();
newItem.name = 'New Customer';

// commit the new item so the view can be refreshed
view.commitNew();
```

You can also add new items by pushing them into the **sourceCollection** and then calling the **refresh** method. The main advantage of **addNew** is in user-interactive scenarios (like adding new items in a data grid), because it gives users the ability to cancel the add operation. It also prevents the new item from being sorted or filtered out of view until the add operation is committed.

Inherited From
CollectionView
Returns
any

beginUpdate

```
beginUpdate(): void
```

Suspend refreshes until the next call to **endUpdate**.

Inherited From
CollectionView
Returns
void

cancelEdit

```
cancelEdit(): void
```

Ends the current edit transaction and, if possible, restores the original value to the item.

Inherited From
`CollectionView`
Returns
void

cancelNew

```
cancelNew(): void
```

Ends the current add transaction and discards the pending new item.

Inherited From
`CollectionView`
Returns
void

clearChanges

```
clearChanges(): void
```

Clears all changes by removing all items in the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

Call this method after committing changes to the server or after refreshing the data from the server.

Inherited From
`CollectionView`
Returns
void

commitEdit

```
commitEdit(): void
```

Override **commitEdit** to modify the item in the database.

Inherited From
ODataCollectionView
Returns
void

commitNew

```
commitNew(): void
```

Override **commitNew** to add the new item to the database.

Inherited From
ODataCollectionView
Returns
void

contains

```
contains(item: any): boolean
```

Returns a value indicating whether a given item belongs to this view.

Parameters

- **item**: **any**
  Item to seek.

Inherited From
CollectionView
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The collection will not be refreshed until the function finishes. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
`CollectionView`
Returns
void

editItem

```
editItem(item: any): void
```

Begins an edit transaction of the specified item.

Parameters

- **item**: **any**
  Item to be edited.

Inherited From
`CollectionView`
Returns
void

endUpdate

```
endUpdate(): void
```

Resume refreshes suspended by a call to **beginUpdate**.

Inherited From
CollectionView
Returns
void

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

Calculates an aggregate value for the items in this collection.

Parameters

- **aggType**: **Aggregate**
  Type of aggregate to calculate.
- **binding**: **string**
  Property to aggregate on.
- **currentPage**: **boolean** OPTIONAL
  Whether to include only items on the current page.

Inherited From
CollectionView
Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
`CollectionView`
Returns
boolean

load

```
load(): void
```

Loads or re-loads the data from the OData source.

Inherited From
`ODataCollectionView`
Returns
void

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

Sets the specified item to be the current item in the view.

Parameters

- **item**: **any**
  Item that will become current.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

Sets the first item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

Sets the last item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

Sets the item after the current item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

Sets the item at the specified index in the view as the current item.

Parameters

- **index**: **number**
  Index of the item that will become current.

Inherited From
CollectionView
Returns
boolean

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

Sets the item before the current item in the view as the current item.

Inherited From
CollectionView
Returns
boolean

moveToFirstPage

```
moveToFirstPage(): boolean
```

Sets the first page as the current page.

Inherited From
CollectionView
Returns
boolean

moveToLastPage

```
moveToLastPage(): boolean
```

Sets the last page as the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToNextPage

```
moveToNextPage(): boolean
```

Moves to the page after the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToPage

```
moveToPage(index: number): boolean
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index of the page to move to.

Inherited From
`CollectionView`
Returns
boolean

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

Moves to the page before the current page.

Inherited From
CollectionView
Returns
boolean

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
CollectionView
Returns
void

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

Raises the **currentChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
CollectionView
Returns
void

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

Raises the **currentChanging** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
CollectionView
Returns
boolean

onError

```
onError(e: RequestErrorEventArgs): boolean
```

Raises the **error** event.

By default, errors throw exceptions and trigger a data refresh. If you want to prevent this behavior, set the **cancel** parameter to true in the event handler.

Parameters

- **e**: **RequestErrorEventArgs**
  RequestErrorEventArgs that contains information about the error.

Inherited From
ODataCollectionView
Returns
boolean

onLoaded

```
onLoaded(e?: EventArgs): void
```

Raises the **loaded** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ODataCollectionView
Returns
void

onLoading

```
onLoading(e?: EventArgs): void
```

Raises the **loading** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ODataCollectionView
Returns
void

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

Raises the **pageChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
CollectionView
Returns
void

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

Raises the **pageChanging** event.

Parameters

- **e**: **PageChangingEventArgs**
  `PageChangingEventArgs` that contains the event data.

Inherited From
`ODataCollectionView`
Returns
boolean

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

Raises the **sourceCollectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`CollectionView`
Returns
void

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

Raises the **sourceCollectionChanging** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
CollectionView
Returns
boolean

refresh

```
refresh(): void
```

Re-creates the view using the current sort, filter, and group parameters.

Inherited From
CollectionView
Returns
void

remove

```
remove(item: any): void
```

Override **remove** to remove the item from the database.

Parameters

- **item**: **any**
  Item to be removed from the database.

Inherited From
ODataCollectionView
Returns
void

removeAt

```
removeAt(index: number): void
```

Removes the item at the specified index from the collection.

Parameters

- **index**: **number**
  Index of the item to be removed from the collection. The index is relative to the view, not to the source collection.

Inherited From
`CollectionView`
Returns
void

setWindow

```
setWindow(start: number, end: number): void
```

Sets the data window to ensure a range of records are loaded into the view.

Parameters

- **start**: **number**
  Index of the first item in the data window.

- **end**: **number**
  Index of the last item in the data window.

Returns
void

updateFilterDefinition

```
updateFilterDefinition(filterProvider: any): void
```

Updates the filter definition based on a known filter provider such as the **`FlexGridFilter`**.

Parameters

- **filterProvider**: **any**
  Known filter provider, typically an instance of a **`FlexGridFilter`**.

Inherited From
`ODataCollectionView`
Returns
void

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`CollectionView`
Arguments
`NotifyCollectionChangedEventArgs`

currentChanged

Occurs after the current item changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

currentChanging

Occurs before the current item changes.

Inherited From
`CollectionView`
Arguments
`CancelEventArgs`

error

Occurs when there is an error reading or writing data.

Inherited From
`ODataCollectionView`
Arguments
`RequestErrorEventArgs`

loaded

Occurs when the **`ODataCollectionView`** finishes loading data.

Inherited From
`ODataCollectionView`
Arguments
`EventArgs`

loading

Occurs when the **`ODataCollectionView`** starts loading data.

Inherited From
`ODataCollectionView`
Arguments
`EventArgs`

pageChanged

Occurs after the page index changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

pageChanging

Occurs before the page index changes.

Inherited From
`CollectionView`
Arguments
`PageChangingEventArgs`

sourceCollectionChanged

Occurs after the value of the **`sourceCollection`** property changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

sourceCollectionChanging

Occurs before the value of the **`sourceCollection`** property changes.

Inherited From
`CollectionView`
Arguments
`CancelEventArgs`

wijmo.xlsx Module

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

The module has a dependency on the **JSZip** library which can be referenced as follows:

- In order to invoke the synchronous save and load methods, JSZip2 library should be referenced in html page with the markup like this:

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js"></script>
```

- In order to invoke the asynchronous save and load methods, JSZip3 library should be referenced in html page with the markup like this:

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/3.1.3/jszip.min.js"></script>
```

Classes

- DefinedName
- Workbook
- WorkbookBorder
- WorkbookBorderSetting
- WorkbookCell
- WorkbookColumn
- WorkbookFill
- WorkbookFont
- WorkbookFrozenPane
- WorkbookRow
- WorkbookStyle
- WorkSheet

Interfaces

- IDefinedName
- ITableAddress
- IWorkbook
- IWorkbookBorder
- IWorkbookBorderSetting
- IWorkbookCell
- IWorkbookColumn
- IWorkbookFill
- IWorkbookFont
- IWorkbookFrozenPane
- IWorkbookRow
- IWorkbookStyle
- IWorkSheet

Enums

- BorderStyle
- HAlign
- VAlign

DefinedName Class

File
wijmo.xlsx.js
Module
wijmo.xlsx
Implements
IDefinedName

Represents the Workbook Object Model Defined Name item definition.

Constructor

- constructor

Properties

- name
- sheetName
- value

Constructor

constructor

```
constructor(): DefinedName
```

Initializes a new instance of the **DefinedName** class.

Returns
DefinedName

Properties

name

The name of the defined name item.

Type
string

sheetName

Indicates the defined name item works in which sheet. If omitted, the defined name item works in workbook

Type
string

value

The value of the defined name item. The value could be a formula, a string constant or a cell range. For e.g. "Sum(1, 2, 3)", "test" or "sheet1!A1:B2"

Type
any

Workbook Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbook

Represents an Excel workbook.

Constructor

- constructor

Properties

- activeWorksheet
- application
- colorThemes
- company
- created
- creator
- definedNames
- lastModifiedBy
- modified
- reservedContent
- sheets
- styles

Methods

- fromXlsxFormat
- load
- loadAsync
- save
- saveAsync
- tableAddress
- toXlsxDateFormat
- toXlsxNumberFormat
- xlsxAddress

Constructor

constructor

```
constructor(): Workbook
```

Initializes a new instance of the **Workbook** class.

Returns
Workbook

Properties

activeWorksheet

Gets or sets the index of the active sheet in the xlsx file.

Type
number

application

Gets or sets the name of application that generated the file that appears in the file properties.

Type
string

colorThemes

Gets the color of the workbook themes.

Type
string[]

company

Gets or sets the name of company that generated the file that appears in the file properties.

Type
string

created

Gets or sets the creation time of the xlsx file.

Type
Date

creator

Gets or sets the creator of the xlsx file.

Type
string

definedNames

Gets the defined name items of the workbook.

Type
`DefinedName`[]

lastModifiedBy

Gets or sets the last modifier of the xlsx file.

Type
string

modified

Gets or sets the last modified time of the xlsx file.

Type
Date

reservedContent

Gets or sets the reserved content from xlsx file that flexgrid or flexsheet doesn't support yet.

Type
any

sheets

Gets the WorkSheet array of the workbook.

Type
`WorkSheet[]`

styles

Gets the styles table of the workbook.

Type
`WorkbookStyle[]`

Methods

STATIC fromXlsxFormat

```
fromXlsxFormat(xlsxFormat: string): string[]
```

Converts the xlsx multi-section format string to an array of corresponding wijmo formats.

Parameters

- **xlsxFormat**: **string**
  The Excel format string, that may contain multiple format sections separated by a semicolon.

Returns
string[]

load

```
load(base64: string): void
```

Loads from base-64 string or data url. This method works with JSZip 2.5.

For example:

```
// This sample opens an xlsx file chosen from Open File
// dialog and creates a workbook instance to load the file.


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>


// JavaScript
var workbook, // receives imported IWorkbook
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        workbook,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
           workbook = new wijmo.xlsx.Workbook(),
           workbook.load(reader.result);
        };
        reader.readAsDataURL(file);
    }
}
```

Parameters

- **base64**: **string**

  The base-64 string that contains the xlsx file content.

Returns
void

loadAsync

```
loadAsync(base64: string, onLoaded?: (workbook: Workbook), onError?: (reason?: any)): void
```

Loads from base-64 string or data url asynchronously. This method works with JSZip 3.0.

Parameters

- **base64**: **string**
  base64 string that contains the xlsx file content.

- **onLoaded**: **(workbook: Workbook)** OPTIONAL
  This callback provides an approach to get an instance of the loaded workbook. Since this method is an asynchronous method, user is not able to get instance of the loaded workbook immediately. User has to get the instance through this callback. This has a single parameter, instance of the loaded workbook. It will be passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when loading. This has a single parameter, the failure reason. Return value is be passed to user, if he wants to catch the load failure reason.

  For example:

```
workbook.loadAsync(base64, function (workbook) {

     // User can access the loaded workbook instance in this callback.
     var app = worksheet.application ;
     ...
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of load failure is ' + reason);
});
```

Returns
void

save

```
save(fileName?: string): string
```

Saves the book to a file and returns a base-64 string representation of the book. This method works with JSZip 2.5.

For example, this sample creates an xlsx file with a single cell:

```
function exportXlsx(fileName) {
    var book = new wijmo.xlsx.Workbook(),
        sheet = new wijmo.xlsx.WorkSheet(),
        bookRow = new wijmo.xlsx.WorkbookRow(),
        bookCell = new wijmo.xlsx.WorkbookCell();
    bookCell.value = 'Hello, Excel!';
    bookRow.cells.push(bookCell);
    sheet.rows.push(bookRow);
    book.sheets.push(sheet);
    book.save(fileName);
}
```

The file name is optional. If not provided, the method still returns a base-64 string representing the book. This string can be used for further processing on the client or on the server.

Parameters

- **fileName**: **string** OPTIONAL
  Name of the xlsx file to save.

Returns
string

saveAsync

```
saveAsync(fileName?: string, onSaved?: (base64?: string), onError?: (reason?: any)): void
```

Saves the book to a file asynchronously. This method works with JSZip 3.0.

Parameters

- **fileName**: **string** OPTIONAL
  Name of the xlsx file to save.

- **onSaved**: **(base64?: string)** OPTIONAL
  This callback provides an approach to get the base-64 string that represents the content of the saved workbook. Since this method is an asynchronous method, user does not get the base-64 string immediately. User has to get the base-64 string via this callback. This has a single parameter, the base-64 string of the saved workbook. It will be passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when saving. This has a single parameter, the failure reason. Return value will be passed to user, if he wants to catch the save failure reason.

  For example:

```
workbook.saveAsync('', function (base64){

     // User can access the base64 string in this callback.
     document.getElementByID('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64,' + base64;
}, function (reason){

     // User can catch the failure reason in this callback.
     console.log('The reason of save failure is ' + reason);
});
```

Returns
void

STATIC tableAddress

```
tableAddress(xlsxIndex: string): ITableAddress
```

Convert Excel's alphanumeric cell, row or column index to the zero-based row/column indices pair.

Parameters

- **xlsxIndex**: **string**
  The alphanumeric Excel index that may include alphabetic A-based column index and/or numeric 1-based row index, like "D15", "D" or "15". The alphabetic column index can be in lower or upper case.

Returns
`ITableAddress`

STATIC toXlsxDateFormat

```
toXlsxDateFormat(format: string): string
```

Converts the wijmo date format to Excel format.

Parameters

- **format**: **string**
  The wijmo date format.

Returns
string

STATIC toXlsxNumberFormat

```
toXlsxNumberFormat(format: string): string
```

Converts the wijmo number format to xlsx format.

Parameters

- **format**: **string**
  The wijmo number format.

Returns
string

STATIC xlsxAddress

```
xlsxAddress(row: number, col: number, absolute?: boolean, absoluteCol?: boolean): string
```

Converts zero-based cell, row or column index to Excel alphanumeric representation.

Parameters

- **row**: **number**
  The zero-based row index or a null value if only column index is to be converted.
- **col**: **number**
  The zero-based column index or a null value if only row index is to be converted.
- **absolute**: **boolean** OPTIONAL
  True value indicates that absolute indices is to be returned for both, row and column (like D7). The **absoluteCol** parameter allows to redefine this value for the column index.
- **absoluteCol**: **boolean** OPTIONAL
  True value indicates that column index is absolute.

Returns
string

WorkbookBorder Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookBorder

Represents the Workbook Object Model border definition.

Constructor

- constructor

Properties

- bottom
- diagonal
- left
- right
- top

Constructor

constructor

```
constructor(): WorkbookBorder
```

Initializes a new instance of the **`WorkbookBorder`** class.

Returns
WorkbookBorder

Properties

bottom

Gets or sets the bottom border setting.

Type
`WorkbookBorderSetting`

diagonal

Gets or sets the diagonal border setting.

Type
`WorkbookBorderSetting`

left

Gets or sets the left border setting.

Type
`WorkbookBorderSetting`

right

Gets or sets the right border setting.

Type
`WorkbookBorderSetting`

top

Gets or sets the top border setting.

Type
`WorkbookBorderSetting`

WorkbookBorderSetting Class

File
wijmo.xlsx.js
Module
wijmo.xlsx
Implements
IWorkbookBorderSetting

Represents the Workbook Object Model background setting definition.

Constructor

- constructor

Properties

- color
- style

Constructor

constructor

```
constructor(): WorkbookBorderSetting
```

Initializes a new instance of the **WorkbookBorderSetting** class.

Returns
WorkbookBorderSetting

Properties

color

Gets or sets the border color.

For export, the color can be specified in any valid HTML format like 6-character dash notation or rgb/rgba/hsl/hsla functional form. In case of rgba/hsla representations, specified alpha channel value is ignored.

For import, a value is always represented in the HTML 6-character dash notation, for example, "#afbfcf".

Type
any

style

Gets or sets the border type.

Type
`BorderStyle`

WorkbookCell Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookCell

Represents the Workbook Object Model cell definition.

Constructor

- constructor

Properties

- colSpan
- formula
- isDate
- rowSpan
- style
- value

Constructor

constructor

```
constructor(): WorkbookCell
```

Initializes a new instance of the **`WorkbookCell`** class.

Returns
WorkbookCell

Properties

colSpan

Gets or sets the colSpan setting of cell.

Type
number

formula

Gets or sets the formula of cell.

Type
string

isDate

Indicates whether the cell value is date or not.

Type
boolean

rowSpan

Gets or sets the rowSpan setting of cell.

Type
number

style

Gets or sets the style of cell.

Type
`WorkbookStyle`

value

Gets or sets the cell value.

The type of the value can be String, Number, Boolean or Date.

Type
any

WorkbookColumn Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookColumn

Represents the Workbook Object Model column definition.

Constructor

- constructor

Properties

- autoWidth
- style
- visible
- width

Constructor

constructor

```
constructor(): WorkbookColumn
```

Initializes a new instance of the **`WorkbookColumn`** class.

Returns
WorkbookColumn

Properties

autoWidth

Gets or sets a value indicating whether the column width is automatically adjusted to fit the content of its cells.

Type
boolean

style

Gets or sets the column style.

The property defines the style for all cells in the column, and can be overridden by the specific cell styles.

Type
`WorkbookStyle`

visible

Gets or sets the column visibility.

Type
boolean

width

Gets or sets the width of the column in device-independent (1/96th inch) pixels or characters.

The numeric value defines the width in pixels. On import, the widths are always expressed in pixels.

The string value which is a number with the 'ch' suffix, for example '10ch', defines the width in characters. It has the same meaning as the column width defined through Excel UI. The width can be specified in characters for the export operations only.

If width is not specified, then the default width is applied.

Type
any

WorkbookFill Class

File
wijmo.xlsx.js
Module
wijmo.xlsx
Implements
IWorkbookFill

Represents the Workbook Object Model background fill definition.

Constructor

- constructor

Properties

- color

Constructor

constructor

```
constructor(): WorkbookFill
```

Initializes a new instance of the **WorkbookFill** class.

Returns
WorkbookFill

Properties

color

Gets or sets the fill color.

For export, the color can be specified in any valid HTML format like 6-character dash notation or rgb/rgba/hsl/hsla functional form. In case of rgba/hsla representations, specified alpha channel value is ignored.

For import, a value is always represented in the HTML 6-character dash notation, for example, "#afbfcf".

Type
any

WorkbookFont Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookFont

Represents the Workbook Object Model font definition.

Constructor

- constructor

Properties

- bold
- color
- family
- italic
- size
- underline

Constructor

constructor

```
constructor(): WorkbookFont
```

Initializes a new instance of the **`WorkbookFont`** class.

Returns
WorkbookFont

Properties

bold

Indicates whether the current font is bold.

Type
boolean

color

Gets or sets the font color.

For export, the color can be specified in any valid HTML format like 6-character dash notation or rgb/rgba/hsl/hsla functional form. In case of rgba/hsla representations, specified alpha channel value is ignored.

For import, a value is always represented in the HTML 6-character dash notation, for example, "#afbfcf".

Type
any

family

Gets or sets the font family name.

Type
string

italic

Indicates whether the current font has the italic style applied.

Type
boolean

size

Gets or sets the font size in device-independent (1/96th inch) pixels.

Type
number

underline

Indicates whether the current font is underlined.

Type
boolean

WorkbookFrozenPane Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookFrozenPane

Workbook frozen pane definition

Constructor

- constructor

Properties

- columns
- rows

Constructor

constructor

```
constructor(): WorkbookFrozenPane
```

Initializes a new instance of the **WorkbookFrozenPane** class.

Returns
WorkbookFrozenPane

Properties

columns

Gets or sets the number of frozen columns.

Type
number

rows

Gets or sets the number of frozen rows.

Type
number

WorkbookRow Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookRow

Represents the Workbook Object Model row definition.

Constructor

- constructor

Properties

- cells
- collapsed
- groupLevel
- height
- style
- visible

Constructor

constructor

```
constructor(): WorkbookRow
```

Initializes a new instance of the **WorkbookRow** class.

Returns
WorkbookRow

Properties

cells

Gets or sets an array of cells in the row.

Each **`WorkbookCell`** object in the array describes a cell at the corresponding position in the row, i.e. a cell with index 0 pertains to column with index A, a cell with index 1 defines cell pertaining to column with index B, and so on. If a certain cell has no definition (empty) in xlsx file, then corresponding array element is undefined for both export and import operations.

Type
`WorkbookCell`[]

collapsed

Indicating if the row is in the collapsed outline state.

Type
boolean

groupLevel

Gets or sets the group level of the row.

Type
number

height

Gets or sets the row height in device-independent (1/96th inch) pixels.

If height is not specified, then the default height is applied.

Type
number

style

Gets or sets the row style.

The property defines the style for all cells in the row, and can be overridden by the specific cell styles.

Type
`WorkbookStyle`

visible

Gets or sets the row visibility.

Type
boolean

WorkbookStyle Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkbookStyle

Represents the Workbook Object Model style definition used to style Excel cells, columns and rows.

Constructor

- constructor

Properties

- basedOn
- borders
- fill
- font
- format
- hAlign
- indent
- vAlign
- wordWrap

Constructor

constructor

```
constructor(): WorkbookStyle
```

Initializes a new instance of the **`WorkbookStyle`** class.

Returns
WorkbookStyle

Properties

basedOn

Defines the base style that this style inherits.

This property is applicable for the export operations only. The style gets all the properties defined in the base style, and can override or augment them by setting its own properties.

Type
`WorkbookStyle`

borders

Gets or sets the border setting.

Type
`WorkbookBorder`

fill

Gets or sets the background setting.

Type
`WorkbookFill`

font

Gets or sets the font of style.

Type
`WorkbookFont`

format

Cell value format, defined using Excel format syntax.

The description of Excel format syntax can be found **here**.

You may use the **toXlsxNumberFormat** and **toXlsxDateFormat** static functions of the `Workbook` class to convert from .Net (`Globalize`) format to Excel format.

Type
string

hAlign

Gets or sets the horizontal alignment of text.

Type
`HAlign`

indent

Gets or sets the indent setting of style.

Type
number

vAlign

Gets or sets the vertical alignment of text.

Type
`VAlign`

wordWrap

Gets or sets the word wrap setting of row.

Type
boolean

WorkSheet Class

File
`wijmo.xlsx.js`
Module
wijmo.xlsx
Implements
IWorkSheet

Represents the Workbook Object Model sheet definition that includes sheet properties and data.

The sheet cells are stored in row objects and are accessible using JavaScript expressions like **sheet.rows[i].cells[j]**.

Constructor

- constructor

Properties

- columns
- frozenPane
- name
- rows
- style
- summaryBelow
- visible

Constructor

constructor

```
constructor(): WorkSheet
```

Initializes a new instance of the **WorkSheet** class.

Returns
WorkSheet

Properties

columns

Gets or sets an array of sheet columns definitions.

Each **WorkbookColumn** object in the array describes a column at the corresponding position in xlsx sheet, i.e. the column with index 0 corresponds to xlsx sheet column with index A, object with index 1 defines sheet column with index B, and so on. If certain column has no description in xlsx file, then corresponding array element is undefined for both export and import operations.

If **WorkbookColumn** object in the array doesn't specify the **width** property value, then the default column width is applied.

Type
WorkbookColumn[]

frozenPane

Gets or sets the **WorkbookFrozenPane** settings.

Type
WorkbookFrozenPane

name

Gets or sets the sheet name.

Type
string

rows

Gets an array of sheet rows definition.

Each **WorkbookRow** object in the array describes a row at the corresponding position in xlsx sheet, i.e. the row with index 0 corresponds to excel sheet row with index 1, object with index 1 defines sheet row with index 2, and so on. If certain row has no properties and data in xlsx file, then corresponding array element is undefined for both export and import operations.

If **WorkbookRow** object in the array doesn't specify the **height** property value, then the default row height is applied.

Type
WorkbookRow[]

style

Gets or sets the row style.

The property defines the style for all cells in the worksheet, and can be overridden by the specific cell styles.

Type
`WorkbookStyle`

summaryBelow

Gets or sets a value indicating whether summary rows appear below or above detail rows.

Type
boolean

visible

Gets or sets the worksheet visibility.

Type
boolean

IDefinedName Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Defined name definition.

Properties

- name
- sheetName
- value

Properties

- **name**

The name of the defined name item.

Type
string

- **sheetName**

Indicates the defined name item works in which sheet. If omitted, the defined name item works in workbook.

Type
string

- **value**

The value of the defined name item. The value could be a formula, a string constant or a cell range. For e.g. "Sum(1, 2, 3)", "test" or "sheet1!A1:B2"

Type
any

ITableAddress Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Defines a cell index with zero-based row and column components, as well as the properties indicating whether the index component is absolute (for example: "$D") or relative (for example: "D").

Properties

- absCol
- absRow
- col
- row

Properties

absCol

Indicates whether the original column index is absolute (for example: "$D") or relative (for example: "D").

Type
boolean

absRow

Indicates whether the original row index is absolute (for example: "$15") or relative (for example: "15").

Type
boolean

col

A zero-based column index.

Type
number

row

A zero-based row index.

Type
number

IWorkbook Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents an Excel Workbook. This interface is the root of the Excel Workbook Object Model (WOM) which provides a way to define properties and data stored in xlsx file.

To create an xlsx file, create a **Workbook** object and populate them with **WorkSheet**, **WorkbookColumn**, **WorkbookRow**, and **WorkbookCell** objects.

To save xlsx files, use the **save** method which can save the book to a file or return it as a base-64 string.

To load existing xlsx files, use the **load** method which will populate the book.

Properties

- activeWorksheet
- application
- colorThemes
- company
- created
- creator
- definedNames
- lastModifiedBy
- modified
- reservedContent
- sheets
- styles

Properties

activeWorksheet

Index of the active sheet in the xlsx file.

Type
number

application

Name of the application that generated the file that appears in the file properties.

Type
string

colorThemes

The color of the workbook themes.

Type
string[]

company

Name of the company that generated the file that appears in the file properties.

Type
string

created

Creation time of the xlsx file.

Type
Date

creator

Creator of the xlsx file.

Type
string

definedNames

The array of the defined name items.

Type
`IDefinedName`[]

lastModifiedBy

Last modifier of the xlsx file.

Type
string

modified

Last modified time of the xlsx file.

Type
Date

reservedContent

The reserved content for the workbook.

Type
any

sheets

Defines an array of Excel Workbook sheets.

Type
`IWorkSheet`[]

styles

Styles table of the workbook.

Type
`IWorkbookStyle`[]

IWorkbookBorder Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Workbook cell outline definition.

Properties

- bottom
- diagonal
- left
- right
- top

Properties

bottom

Bottom border setting.

Type
`IWorkbookBorderSetting`

diagonal

Diagonal border setting.

Type
`IWorkbookBorderSetting`

left

Left border setting.

Type
`IWorkbookBorderSetting`

right

Right border setting.

Type
`IWorkbookBorderSetting`

top

Top border setting.

Type
`IWorkbookBorderSetting`

IWorkbookBorderSetting Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Border style definition

Properties

- color
- style

Properties

- **color**

Border color.

For export, the color can be specified in any valid HTML format like 6-character dash notation or rgb/rgba/hsl/hsla functional form. In case of rgba/hsla representations, specified alpha channel value is ignored.

For import, a value is always represented in the HTML 6-character dash notation, for example, "#afbfcf".

Type
any

- **style**

Border type.

Type
`BorderStyle`

IWorkbookCell Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model cell definition.

Properties

- colSpan
- formula
- isDate
- rowSpan
- style
- value

Properties

colSpan

Cell colSpan setting

Type
number

formula

Cell formula

Type
string

isDate

Indicates whether the cell value is date or not.

Type
boolean

rowSpan

Cell rowSpan setting

Type
number

style

Cell style

Type
`IWorkbookStyle`

value

Gets or sets the cell value.

The type of the value can be String, Number, Boolean or Date.

Type
any

IWorkbookColumn Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model column definition.

Properties

- autoWidth
- style
- visible
- width

Properties

autoWidth

Gets or sets a value indicating whether the column width is automatically adjusted to fit the content of its cells.

Type
boolean

style

Gets or sets the column style.

The property defines the style for all cells in the column, and can be overridden by the specific cell styles.

Type
`IWorkbookStyle`

visible

Gets or sets the column visibility.

Type
boolean

width

Gets or sets the width of the column in device-independent (1/96th inch) pixels or characters.

The numeric value defines the width in pixels. On import, the widths are always expressed in pixels.

The string value which is a number with the 'ch' suffix, for example '10ch', defines the width in characters. It has the same meaning as the column width defined through Excel UI. The width can be specified in characters for the export operations only.

If width is not specified, then the default width is applied.

Type
any

IWorkbookFill Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model background fill definition.

Properties

- color

Properties

- **color**

Gets or sets the fill color.

For export, the color can be specified in any valid HTML format like 6-character dash notation or rgb/rgba/hsl/hsla functional form. In case of rgba/hsla representations, specified alpha channel value is ignored.

For import, a value is always represented in the HTML 6-character dash notation, for example, "#afbfcf".

Type
any

IWorkbookFont Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model font definition.

Properties

- bold
- color
- family
- italic
- size
- underline

Properties

bold

Gets or sets a value indicating whether this font is bold.

Type
boolean

color

Gets or sets the font color.

For export, the color can be specified in any valid HTML format like 6-character dash notation or rgb/rgba/hsl/hsla functional form. In case of rgba/hsla representations, specified alpha channel value is ignored.

For import, a value is always represented in the HTML 6-character dash notation, for example, "#afbfcf".

Type
any

family

Gets or sets the font family name.

Type
string

italic

Gets or sets a value indicating whether this font has the italic style applied.

Type
boolean

size

Gets or sets the font size in device-independent (1/96th inch) pixels.

Type
number

underline

Gets or sets a value indicating whether this font is underlined.

Type
boolean

IWorkbookFrozenPane Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Workbook frozen pane definition

Properties

- columns
- rows

Properties

- **columns**

Gets or sets the number of frozen columns.

Type
number

- **rows**

Gets or sets the number of frozen rows.

Type
number

IWorkbookRow Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model row definition.

Properties

- cells
- collapsed
- groupLevel
- height
- style
- visible

Properties

cells

Gets or sets an array of cells in the row.

Each **`IWorkbookCell`** object in the array describes a cell at the corresponding position in the row, i.e. cell with index 0 pertains to column with index A, cell with index 1 defines cell pertaining to column with index B, and so on. If a certain cell has no definition (empty) in xlsx file, then corresponding array element is undefined for both export and import operations.

Type
`IWorkbookCell`[]

collapsed

TBD: Indicating if the row is in the collapsed outline state.

Type
boolean

groupLevel

Gets or sets the group level of the row.

Type
number

height

Gets or sets the row height in device-independent (1/96th inch) pixels.

If height is not specified, then the default height is applied.

Type
number

style

Gets or sets the row style.

The property defines the style for all cells in the row, and can be overridden by the specific cell styles.

Type
`IWorkbookStyle`

visible

Gets or sets the row visibility.

Type
boolean

IWorkbookStyle Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model style definition used to style Excel cells, columns and rows.

Properties

Properties

basedOn

Defines the base style that this style inherits.

This property is applicable for export operations only. The style gets all the properties defined in the base style, and can override or augment them by setting its own properties.

Type
`IWorkbookStyle`

borders

Cell outline setting.

Type
`IWorkbookBorder`

fill

Cells background.

Type
`IWorkbookFill`

font

Gets or sets the font properties.

Type
`IWorkbookFont`

format

Cell value format, defined using Excel format syntax.

The description of Excel format syntax can be found **here**.

You may use the **toXlsxNumberFormat** and **toXlsxDateFormat** static functions of the `Workbook` class to convert from .Net (`Globalize`) format to Excel format.

Type
string

hAlign

Gets or sets the horizontal alignment of a text.

Type
`HAlign`

indent

Text indent. It is an integer value, where an increment of 1 represents 3 spaces.

Type
number

vAlign

Gets or sets the vertical alignment of a text.

Type
`VAlign`

wordWrap

Word wrap setting.

Type
boolean

IWorkSheet Interface

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Represents the Workbook Object Model sheet definition that includes sheet properties and data.

The sheet cells are stored in row objects and are accessible using JavaScript expressions like **sheet.rows[i].cells[j]**.

Properties

- columns
- frozenPane
- name
- rows
- style
- summaryBelow
- visible

Properties

columns

Gets or sets an array of sheet columns definitions.

Each **`IWorkbookColumn`** object in the array describes a column at the corresponding position in xlsx sheet, i.e. column with index 0 corresponds to xlsx sheet column with index A, object with index 1 defines sheet column with index B, and so on. If certain column has no description in xlsx file, then corresponding array element is undefined for both export and import operations.

If **`IWorkbookColumn`** object in the array doesn't specify the **width** property value, then the default column width is applied.

Type
`IWorkbookColumn`[]

frozenPane

Gets or sets the frozen pane settings.

Type
`IWorkbookFrozenPane`

name

Gets or sets the sheet name.

Type
string

rows

Gets or sets an array of sheet rows definition.

Each **`IWorkbookRow`** object in the array describes a row at the corresponding position in xlsx sheet, i.e. row with index 0 corresponds to xlsx sheet row with index A, object with index 1 defines sheet row with index B, and so on. If certain row has no description in xlsx file, then corresponding array element is undefined for both export and import operations.

If **`IWorkbookRow`** object in the array doesn't specify the **height** property value, then the default row height is applied.

Type
`IWorkbookRow`[]

style

Gets or sets the sheet style.

The property defines the style for all cells in the worksheet, and can be overridden by the specific cell styles.

Type
`IWorkbookStyle`

summaryBelow

Gets or sets a value indicating whether summary rows appear below or above detail rows.

Type
boolean

visible

Gets or sets the worksheet visibility.

Type
boolean

BorderStyle Enum

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Border line style

Members

Name	Value	Description
None	0	No border
Thin	1	Thin border
Medium	2	Medium border
Dashed	3	Dashed border
Dotted	4	Dotted border
Thick	5	Thick line border
Double	6	Double line border
Hair	7	Hair line border
MediumDashed	8	Medium dashed border
ThinDashDotted	9	Thin dash dotted border
MediumDashDotted	10	Medium dash dotted border
ThinDashDotDotted	11	Thin dash dot dotted border
MediumDashDotDotted	12	Medium dash dot dotted border
SlantedMediumDashDotted	13	Slanted medium dash dotted border

HAlign Enum

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Defines the Workbook Object Model horizontal text alignment.

Members

Name	Value	Description
General	0	Alignment depends on the cell value type.
Left	1	Text is aligned to the left.
Center	2	Text is centered.
Right	3	Text is aligned to the right.
Fill	4	Text is replicated to fill the whole cell width.
Justify	5	Text is justified.

VAlign Enum

File
`wijmo.xlsx.js`
Module
wijmo.xlsx

Vertical alignment

Members

Name	Value	Description
Top	0	Top vertical alignment
Center	1	Center vertical alignment
Bottom	2	Bottom vertical alignment
Justify	3	Justified vertical alignment

wijmo.pdf Module

File
`wijmo.pdf.js`
Module
wijmo.pdf

Defines the **PdfDocument** class and associated classes.

Classes

- PdfBrush
- PdfDashPattern
- PdfDocument
- PdfDocumentEndedEventArgs
- PdfFont
- PdfGradientBrush
- PdfGradientStop
- PdfLinearGradientBrush
- PdfPageArea
- PdfPaths
- PdfPen
- PdfRadialGradientBrush
- PdfRunningTitle
- PdfRunningTitleDeclarativeContent
- PdfSolidBrush

Interfaces

- IPdfBufferedPageRange
- IPdfDocumentInfo
- IPdfFontAttributes
- IPdfFontFile
- IPdfImageDrawSettings
- IPdfPageMargins
- IPdfPageSettings
- IPdfSvgDrawSettings
- IPdfTextDrawSettings
- IPdfTextMeasurementInfo
- IPdfTextMeasurementSettings
- IPdfTextSettings

Enums

- PdfFillRule
- PdfImageHorizontalAlign
- PdfImageVerticalAlign
- PdfLineCapStyle
- PdfLineJoinStyle
- PdfPageOrientation
- PdfPageSize
- PdfTextHorizontalAlign

Methods

- ptToPx
- pxToPt
- saveBlob

Methods

ptToPx

```
ptToPx(value: number): number
```

Converts a point unit value to a pixel unit value.

Parameters

- **value**: **number**
  The value to convert.

Returns
number

pxToPt

```
pxToPt(value: number): number
```

Converts a pixel unit value to a point unit value.

Parameters

- **value**: **number**
  The value to convert.

Returns
number

saveBlob

```
saveBlob(blob: Blob, fileName: string): void
```

Saves the Blob object as a file.

Parameters

- **blob**: **Blob**
  The Blob object to save.

- **fileName**: **string**
  The name with which the file is saved.

Returns
void

PdfBrush Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Derived Classes
PdfGradientBrush, **PdfSolidBrush**

Represents an abstract class that serves as a base class for all brushes. Instances of any class that derives from this class are used to fill areas and text.

This class is not intended to be instantiated in your code.

Methods

- clone
- equals

Methods

clone

```
clone(): PdfBrush
```

Creates a copy of this **PdfBrush**.

Returns
PdfBrush

equals

```
equals(value: PdfBrush): boolean
```

Determines whether the specified **PdfBrush** instance is equal to the current one.

Parameters

- **value**: **PdfBrush**
  PdfBrush to compare.

Returns
boolean

PdfDashPattern Class

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the dash pattern used to stroke paths.

Constructor

- constructor

Properties

- dash
- gap
- phase

Methods

- clone
- equals

Constructor

constructor

```
constructor(dash?: number, gap?: number, phase?: number): PdfDashPattern
```

Initializes a new instance of the **`PdfDashPattern`** class.

Parameters

- **dash**: **number** OPTIONAL
  The length of alternating dashes, in points.
- **gap**: **number** OPTIONAL
  The length of alternating gaps, in points.
- **phase**: **number** OPTIONAL
  The distance in the dash pattern to start the dash at, in points.

Returns
PdfDashPattern

Properties

dash

Gets or sets the length of alternating dashes, in points. The default value is null which indicates no dash pattern, but a solid line.

Type
number

gap

Gets or sets the length of alternating gaps, in points. The default value is equal to **dash** which indicates that dashes and gaps will have the same length.

Type
number

phase

Gets or sets the distance in the dash pattern to start the dash at, in points. The default value is 0.

Type
number

Methods

clone

```
clone(): PdfDashPattern
```

Creates a copy of this **PdfDashPattern**.

Returns
PdfDashPattern

equals

```
equals(value: PdfDashPattern): boolean
```

Determines whether the specified **PdfDashPattern** instance is equal to the current one.

Parameters

- **value**: **PdfDashPattern**
  PdfDashPattern to compare.

Returns
boolean

PdfDocument Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
PdfPageArea

Represents a PDF document object, based on **PDFKit** JavaScript library.

Constructor

- constructor

Properties

- bufferPages
- compress
- currentPageSettings
- footer
- header
- height
- info
- lineGap
- pageIndex
- pageSettings
- paths
- width
- x
- y

Methods

- addPage
- bufferedPageRange
- dispose
- drawImage
- drawSvg
- drawText
- end
- lineHeight
- measureText
- moveDown
- moveUp
- onEnded
- onPageAdded
- registerFont
- registerFontAsync
- restoreState
- rotate
- saveState
- scale
- setBrush
- setFont
- setPen
- transform
- translate

Events

- ended
- pageAdded

Constructor

constructor

```
constructor(options?: any): PdfDocument
```

Initializes a new instance of the **PdfDocument** class.

Parameters

- **options**: **any** OPTIONAL
  An optional object containing initialization settings.

Returns
PdfDocument

Properties

bufferPages

Gets a value that indicates whether the pages buffering mode is enabled which means that the document's pages can be iterated over using **pageIndex** and **bufferedPageRange**.

This property can be assigned using the **PdfDocument** constructor only. This property can be set to false only if both **header** and **footer** are invisible.

The default value is true.

Type
boolean

compress

Gets a value that indicates whether the document compression is enabled. This property can be assigned using the **PdfDocument** constructor only.

The default value is true.

Type
boolean

currentPageSettings

Gets an object that represents the current page settings (read-only).

Type
IPdfPageSettings

footer

Gets an object that represents a footer, the page area positioned right above the bottom margin.

Type
`PdfRunningTitle`

header

Gets an object that represents a header, the page area positioned right below the top margin.

Type
`PdfRunningTitle`

height

Gets the height of the area, in points.

Inherited From
`PdfPageArea`
Type
number

info

Gets or sets the document information, such as author name, document's creation date and so on.

Type
`IPdfDocumentInfo`

lineGap

Gets or sets the spacing between each line of text, in points.

The default value is 0.

Inherited From
`PdfPageArea`
Type
number

pageIndex

Gets or sets the index of the current page within the buffered pages range.

Use the **`bufferedPageRange`** method to get the range of buffered pages.

Type
number

pageSettings

Gets an object that represents the default page settings for the pages added automatically and for the **`addPage`** method.

Type
`IPdfPageSettings`

paths

Gets an object that provides ability to draw paths.

Inherited From
`PdfPageArea`
Type
`PdfPaths`

width

Gets the width of the area, in points.

Inherited From
`PdfPageArea`
Type
number

x

Gets or sets the X-coordinate (in points) of the current point in the text flow used to draw a text or an image.

Inherited From
`PdfPageArea`
Type
number

y

Gets or sets the Y-coordinate (in points) of the current point in the text flow used to draw a text or an image.

Inherited From
`PdfPageArea`
Type
number

Methods

addPage

```
addPage(settings?: IPdfPageSettings): PdfDocument
```

Adds a new page with the given settings.

If the settings parameter is omitted, then **`pageSettings`** will be used instead.

Parameters

- **settings**: **IPdfPageSettings** OPTIONAL
  Page settings.

Returns
`PdfDocument`

bufferedPageRange

```
bufferedPageRange(): IPdfBufferedPageRange
```

Gets the range of buffered pages.

Returns
IPdfBufferedPageRange

dispose

```
dispose(): void
```

Disposes the document.

Returns
void

drawImage

```
drawImage(url: string, x?: number, y?: number, options?: IPdfImageDrawSettings): PdfPageArea
```

Draws an image in JPG or PNG format with the given options.

If x and y are not defined, then @see:x and @see:y are used instead.

Finally, if the image was drawn in the text flow, the method updates @see:y. Hence, any subsequent text or image starts below this point.

Parameters

- **url**: **string**
  A string containing the URL to get the image from or the data URI containing a base64 encoded image.
- **x**: **number** OPTIONAL
  The x-coordinate of the point to draw the image at, in points.
- **y**: **number** OPTIONAL
  The y-coordinate of the point to draw the image at, in points.
- **options**: **IPdfImageDrawSettings** OPTIONAL
  Determines the image drawing options.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

drawSvg

```
drawSvg(url: string, x?: number, y?: number, options?: IPdfSvgDrawSettings): PdfPageArea
```

Draws a SVG image with the given options.

If x and y are not defined, then @see:x and @see:y are used instead.

The method uses the values of the width and height attributes of the outermost svg element to determine the scale factor according to the options.width and options.height properties. If any of these attributes are omitted then scaling is not performed and the image will be rendered in its original size.

Finally, if the image was drawn in the text flow, the method updates @see:y. Hence, any subsequent text or image starts below this point. The increment value is defined by the options.height property or by the outermost svg element's height attribute, which comes first. If none of them is provided then @see:y will stay unchanged.

The method supports a limited set of SVG features and provided primarily for rendering wijmo 5 chart controls.

Parameters

- **url**: **string**
  A string containing the URL to get the SVG image from or the data URI containing a base64 encoded SVG image.
- **x**: **number** OPTIONAL
  The x-coordinate of the point to draw the image at, in points.
- **y**: **number** OPTIONAL
  The y-coordinate of the point to draw the image at, in points.
- **options**: **IPdfSvgDrawSettings** OPTIONAL
  Determines the SVG image drawing options.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

drawText

```
drawText(text: string, x?: number, y?: number, options?: IPdfTextDrawSettings): IPdfTextMeasurementInfo
```

Draws a string with the given options and returns the measurement information.

If **options.pen**, **options.brush** or **options.font** are omitted, the current document's pen, brush or font are used (see **setPen**, **setBrush**, and **setFont**).

The string is drawn within the rectangular area for which top-left corner, width and height are defined by the x, y, **options.width** and **options.height** values. If x and y are not provided, the **x** and **y** properties are used instead.

The text is wrapped and clipped automatically within the area. If **options.height** is not provided and the text exceeds the bottom body edge, then a new page will be added to accommodate the text.

Finally, the method updates the value of the **x** and **y** properties. Hence, any subsequent text or image starts below this point (depending on the value of **options.continued**).

The measurement result doesn't reflect the fact that text can be split into multiple pages or columns; the text is treated as a single block.

Parameters

- **text**: **string**
  The text to draw.
- **x**: **number** OPTIONAL
  The X-coordinate of the point to draw the text at, in points.
- **y**: **number** OPTIONAL
  The Y-coordinate of the point to draw the text at, in points.
- **options**: **IPdfTextDrawSettings** OPTIONAL
  Determines the text drawing options.

Inherited From
PdfPageArea
Returns
IPdfTextMeasurementInfo

end

```
end(): void
```

Finishes the document rendering.

Returns
void

lineHeight

```
lineHeight(font?: PdfFont): number
```

Gets the line height with a given font.

If font is not specified, then font used in the current document is used.

Parameters

- **font**: **PdfFont** OPTIONAL
  Font to get the line height.

Inherited From
PdfPageArea
Returns
number

measureText

```
measureText(text: string, font?: PdfFont, options?: IPdfTextMeasurementSettings): IPdfTextMeasurementInfo
```

Measures a text with the given font and text drawing options without rendering it.

If font is not specified, then the font used in the current document is used.

The method uses the same text rendering engine as **drawText**, so it is tied up in the same way to @see:x and the right page margin, if options.width is not provided. The measurement result doesn't reflect the fact that text can be split into multiple pages or columns; the text is treated as a single block.

Parameters

- **text**: **string**
  Text to measure.

- **font**: **PdfFont** OPTIONAL
  Font to be applied on the text.

- **options**: **IPdfTextMeasurementSettings** OPTIONAL
  Determines the text drawing options.

Inherited From
PdfPageArea
Returns
IPdfTextMeasurementInfo

moveDown

```
moveDown(lines?: number, font?: PdfFont): PdfPageArea
```

Moves down the @see:y by a given number of lines using the given font or, using the font of current document, if not specified.

Parameters

- **lines**: **number** OPTIONAL
  Number of lines to move down.

- **font**: **PdfFont** OPTIONAL
  Font to calculate the line height.

Inherited From
PdfPageArea
Returns
PdfPageArea

moveUp

```
moveUp(lines?: number, font?: PdfFont): PdfPageArea
```

Moves up the @see:y by a given number of lines using the given font or, using the font of current document, if not specified.

Parameters

- **lines**: **number** OPTIONAL
  Number of lines to move up.

- **font**: **PdfFont** OPTIONAL
  Font to calculate the line height.

Inherited From
PdfPageArea
Returns
PdfPageArea

onEnded

```
onEnded(args: PdfDocumentEndedEventArgs): void
```

Raises the **end** event.

Parameters

- **args**: **PdfDocumentEndedEventArgs**
  A **PdfDocumentEndedEventArgs** object that contains the event data.

Returns
void

onPageAdded

```
onPageAdded(args: EventArgs): void
```

Raises the **pageAdded** event.

Parameters

- **args**: **EventArgs**
  A **EventArgs** object that contains the event data.

Returns
void

registerFont

```
registerFont(font: IPdfFontFile): PdfDocument
```

Registers a font from a source and associates it with a given font family name and font attributes.

Parameters

- **font**: **IPdfFontFile**
  The font to register.

Returns
PdfDocument

registerFontAsync

```
registerFontAsync(font: IPdfFontFile, callback: Function): void
```

Registers a font from a URL asynchronously and associates it with a given font family name and font attributes.

The callback function takes a **`IPdfFontFile`** object as a parameter.

Parameters

- **font**: **IPdfFontFile**
  The font to register.

- **callback**: **Function**
  A callback function which will be called, when the font has been registered.

Returns
void

restoreState

```
restoreState(): PdfDocument
```

Restores the state from the stack and applies it to the graphic context.

Returns
`PdfDocument`

rotate

```
rotate(angle: number, origin?: Point): PdfPageArea
```

Rotates the graphic context clockwise by a specified angle.

Parameters

- **angle**: **number**
  The rotation angle, in degrees.

- **origin**: **Point** OPTIONAL
  The **Point** of rotation, in points. If it is not provided, then the top left corner is used.

Inherited From
PdfPageArea
Returns
PdfPageArea

saveState

```
saveState(): PdfDocument
```

Saves the state of the graphic context (including current pen, brush and transformation state) and pushes it onto stack.

Returns
PdfDocument

scale

```
scale(xFactor: number, yFactor?: number, origin?: Point): PdfPageArea
```

Scales the graphic context by a specified scaling factor.

The scaling factor value within the range [0, 1] indicates that the size will be decreased. The scaling factor value greater than 1 indicates that the size will be increased.

Parameters

- **xFactor**: **number**
  The factor to scale the X dimension.
- **yFactor**: **number** OPTIONAL
  The factor to scale the Y dimension. If it is not provided, it is assumed to be equal to xFactor.
- **origin**: **Point** OPTIONAL
  The **`Point`** to scale around, in points. If it is not provided, then the top left corner is used.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

setBrush

```
setBrush(brushOrColor: any): PdfDocument
```

Sets the default document brush. This brush will be used by the **`fill`**, **`fillAndStroke`** and **`drawText`** methods, if no specific brush is provided.

The brushOrColor argument can accept the following values:

- A **PdfBrush** object.
- A **Color** object or any string acceptable by the **fromString** method. In this case, the **PdfBrush** object with the specified color will be created internally.

Parameters

- **brushOrColor**: **any**
  The brush or color to use.

Returns
`PdfDocument`

setFont

```
setFont(font: PdfFont): PdfDocument
```

Sets the document font. If exact font with given style and weight properties is not found then,

- It tries to search the closest font using **font weight fallback**.
- If still nothing is found, it tries to find the closest font with other style in following order:
  - **'italic'**: 'oblique', 'normal'.
  - **'oblique'**: 'italic', 'normal'.
  - **'normal'**: 'oblique', 'italic'.

Parameters

- **font**: **PdfFont**
  The font object to set.

Returns
`PdfDocument`

setPen

```
setPen(penOrColor: any): PdfDocument
```

Sets the default document pen. This pen will be used by the **`stroke`**, **`fillAndStroke`** and **`drawText`** methods, if no specific pen is provided.

The penOrColor argument can accept the following values:

- A **PdfPen** object.
- A **Color** object or any string acceptable by the **fromString** method. In this case, the **PdfPen** object with the specified color will be created internally.

Parameters

- **penOrColor**: **any**
  The pen or color to use.

Returns
`PdfDocument`

transform

```
transform(a: number, b: number, c: number, d: number, e: number, f: number): PdfPageArea
```

Transforms the graphic context with given six numbers which represents a 3x3 transformation matrix.

A transformation matrix is written as follows:

ab0
cd0
ef 1

Parameters

- **a**: **number**
  Value of the first row and first column.
- **b**: **number**
  Value of the first row and second column.
- **c**: **number**
  Value of the second row and first column.
- **d**: **number**
  Value of the second row and second column.
- **e**: **number**
  Value of the third row and first column.
- **f**: **number**
  Value of the third row and second column.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

translate

```
translate(x: number, y: number): PdfPageArea
```

Translates the graphic context with a given distance.

Parameters

- **x**: **number**
  The distance to translate along the X-axis, in points.

- **y**: **number**
  The distance to translate along the Y-axis, in points.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

Events

ended

Occurs when the document has been rendered.

Arguments
`PdfDocumentEndedEventArgs`

pageAdded

Occurs when a new page is added to the document.

Arguments
`EventArgs`

PdfDocumentEndedEventArgs Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
EventArgs

Provides arguments for the **end** event.

Constructor

- constructor

Properties

- blob
- empty

Constructor

constructor

```
constructor(blob: Blob): PdfDocumentEndedEventArgs
```

Initializes a new instance of the **PdfDocumentEndedEventArgs** class.

Parameters

- **blob**: **Blob**
  A Blob object that contains the document data.

Returns
PdfDocumentEndedEventArgs

Properties

blob

Gets a Blob object that contains the document data.

Type
Blob

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

PdfFont Class

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents a font.

Constructor

- constructor

Properties

- family
- size
- style
- weight

Methods

- clone
- equals

Constructor

constructor

```
constructor(family?: string, size?: number, style?: string, weight?: string): PdfFont
```

Initializes a new instance of the **PdfFont** class.

Parameters

- **family**: **string** OPTIONAL
  The family name of the font.
- **size**: **number** OPTIONAL
  The size of the font.
- **style**: **string** OPTIONAL
  The style of the font.
- **weight**: **string** OPTIONAL
  The weight of the font.

Returns
PdfFont

Properties

family

Gets or sets the family name of the font.

The list of the font family names in the order of preferences, separated by commas. Each font family name can be the one that was registered using the **registerFont** method or the name of one of the PDF standard font families: 'courier', 'helvetica', 'symbol', 'times', 'zapfdingbats' or the superfamily name: 'cursive', 'fantasy', 'monospace', 'serif', 'sans-serif'.

Type
string

size

Gets or sets the size of the font.

Type
number

style

Gets or sets the style of the font.

The following values are supported: 'normal', 'italic', 'oblique'.

Type
string

weight

Gets or sets the weight of the font.

The following values are supported: 'normal', 'bold', '100', '200', '300', '400', '500', '600', '700', '800', '900'.

Type
string

Methods

clone

```
clone(): PdfFont
```

Creates a copy of this **`PdfFont`**.

Returns
`PdfFont`

equals

```
equals(value: PdfFont): boolean
```

Determines whether the specified **PdfFont** instance is equal to the current one.

Parameters

- **value**: **PdfFont**
  PdfFont to compare.

Returns
boolean

PdfGradientBrush Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
PdfBrush
Derived Classes
PdfLinearGradientBrush, **PdfRadialGradientBrush**

Represents an abstract class that serves as a base class for the **PdfLinearGradientBrush** and **PdfRadialGradientBrush** classes.

This class is not intended to be instantiated in your code.

Constructor

- constructor

Properties

- opacity
- stops

Methods

- clone
- equals

Constructor

constructor

```
constructor(stops?: PdfGradientStop[], opacity?: number): PdfGradientBrush
```

Initializes a new instance of the **`PdfGradientBrush`** class.

Parameters

- **stops**: **PdfGradientStop**[] OPTIONAL
  The **`PdfGradientStop`** array to set on this brush.

- **opacity**: **number** OPTIONAL
  The opacity of this brush.

Returns
PdfGradientBrush

Properties

opacity

Gets or sets the opacity of the brush. The value must be in range [0, 1], where 0 indicates that the brush is completely transparent and 1 indicates that the brush is completely opaque. The default value is 1.

Type
number

stops

Gets or sets an array of **`PdfGradientStop`** objects representing a color, offset and opacity within the brush's gradient axis. The default value is an empty array.

Type
`PdfGradientStop`[]

Methods

clone

```
clone(): PdfBrush
```

Creates a copy of this **PdfBrush**.

Inherited From
PdfBrush
Returns
PdfBrush

equals

```
equals(value: PdfGradientBrush): boolean
```

Determines whether the specified **PdfGradientBrush** instance is equal to the current one.

Parameters

- **value**: **PdfGradientBrush**
  PdfGradientBrush to compare.

Returns
boolean

PdfGradientStop Class

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents an object which determines a transition point of a gradient.

Constructor

- constructor

Properties

- color
- offset
- opacity

Methods

- clone
- equals

Constructor

constructor

```
constructor(offset?: number, color?: any, opacity?: number): PdfGradientStop
```

Initializes a new instance of the **`PdfGradientStop`** class.

Parameters

- **offset**: **number** OPTIONAL
  The location of the gradient stop on the gradient axis.
- **color**: **any** OPTIONAL
  The color of the gradient stop. A **`Color`** object or any string acceptable by the **`fromString`** method.
- **opacity**: **number** OPTIONAL
  The opacity of the gradient stop.

Returns
PdfGradientStop

Properties

color

Gets or sets the color of the gradient stop. The default color is black.

Type
`Color`

offset

Gets or sets the location of the gradient stop on gradient axis of the brush. The value must be in range [0, 1], where 0 indicates that the gradient stop is placed at the beginning of the gradient axis, while 1 indicates that the gradient stop is placed at the end of the gradient axis. The default value is 0.

Type
number

opacity

Gets or sets the opacity of the gradient stop. The value must be in range [0, 1], where 0 indicates that the gradient stop is completely transparent, while 1 indicates that the gradient stop is completely opaque. The default value is 1.

Type
number

Methods

clone

```
clone(): PdfGradientStop
```

Creates a copy of this **`PdfGradientStop`**.

Returns
`PdfGradientStop`

equals

```
equals(value: PdfGradientStop): boolean
```

Determines whether the specified **`PdfGradientStop`** instance is equal to the current one.

Parameters

- **value**: **PdfGradientStop**
  `PdfGradientStop` to compare.

Returns
boolean

PdfLinearGradientBrush Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
PdfGradientBrush

Represents a brush used to fill an area with a linear gradient.

Constructor

- constructor

Properties

- opacity
- stops
- x1
- x2
- y1
- y2

Methods

- clone
- equals

Constructor

constructor

```
constructor(x1: number, y1: number, x2: number, y2: number, stops: PdfGradientStop[], opacity?: number): PdfLinearGradientBrush
```

Initializes a new instance of the **`PdfLinearGradientBrush`** class.

Parameters

- **x1**: **number**
  The X-coordinate of the starting point of the linear gradient.
- **y1**: **number**
  The Y-coordinate of the starting point of the linear gradient.
- **x2**: **number**
  The X-coordinate of the ending point of the linear gradient.
- **y2**: **number**
  The Y-coordinate of the ending point of the linear gradient.
- **stops**: **PdfGradientStop**[]
  The **`PdfGradientStop`** array to set on this brush.
- **opacity**: **number** OPTIONAL
  The opacity of this brush.

Returns
PdfLinearGradientBrush

Properties

opacity

Gets or sets the opacity of the brush. The value must be in range [0, 1], where 0 indicates that the brush is completely transparent and 1 indicates that the brush is completely opaque. The default value is 1.

Inherited From
`PdfGradientBrush`
Type
number

- stops

Gets or sets an array of **`PdfGradientStop`** objects representing a color, offset and opacity within the brush's gradient axis. The default value is an empty array.

Inherited From
`PdfGradientBrush`
Type
`PdfGradientStop`[]

- x1

Gets or sets the X-coordinate of the starting point of the linear gradient, in page area coordinates, in points.

Type
number

- x2

Gets or sets the X-coordinate of the ending point of the linear gradient, in page area coordinates, in points.

Type
number

- y1

Gets or sets the Y-coordinate of the starting point of the linear gradient, in page area coordinates, in points.

Type
number

- y2

Gets or sets the Y-coordinate of the ending point of the linear gradient, in page area coordinates, in points.

Type
number

Methods

clone

```
clone(): PdfLinearGradientBrush
```

Creates a copy of this **PdfLinearGradientBrush**.

Returns
PdfLinearGradientBrush

equals

```
equals(value: PdfLinearGradientBrush): boolean
```

Determines whether the specified **PdfLinearGradientBrush** instance is equal to the current one.

Parameters

- **value**: **PdfLinearGradientBrush**
  PdfLinearGradientBrush to compare.

Returns
boolean

PdfPageArea Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Derived Classes
PdfDocument, **PdfRunningTitle**

Represents an area of a page with its own coordinate system, where (0, 0) points to the top-left corner. Provides methods for drawing text, images, paths and transformations.

This class is not intended to be instantiated in your code.

Constructor

- constructor

Properties

- height
- lineGap
- paths
- width
- x
- y

Methods

- drawImage
- drawSvg
- drawText
- lineHeight
- measureText
- moveDown
- moveUp
- rotate
- scale
- transform
- translate

Constructor

constructor

```
constructor(): PdfPageArea
```

Initializes a new instance of the **`PdfRunningTitle`** class.

Returns
PdfPageArea

Properties

height

Gets the height of the area, in points.

Type
number

lineGap

Gets or sets the spacing between each line of text, in points.

The default value is 0.

Type
number

paths

Gets an object that provides ability to draw paths.

Type
`PdfPaths`

width

Gets the width of the area, in points.

Type
number

x

Gets or sets the X-coordinate (in points) of the current point in the text flow used to draw a text or an image.

Type
number

y

Gets or sets the Y-coordinate (in points) of the current point in the text flow used to draw a text or an image.

Type
number

Methods

drawImage

```
drawImage(url: string, x?: number, y?: number, options?: IPdfImageDrawSettings): PdfPageArea
```

Draws an image in JPG or PNG format with the given options.

If x and y are not defined, then @see:x and @see:y are used instead.

Finally, if the image was drawn in the text flow, the method updates @see:y. Hence, any subsequent text or image starts below this point.

Parameters

- **url**: **string**
  A string containing the URL to get the image from or the data URI containing a base64 encoded image.
- **x**: **number** OPTIONAL
  The x-coordinate of the point to draw the image at, in points.
- **y**: **number** OPTIONAL
  The y-coordinate of the point to draw the image at, in points.
- **options**: **IPdfImageDrawSettings** OPTIONAL
  Determines the image drawing options.

Returns
`PdfPageArea`

drawSvg

```
drawSvg(url: string, x?: number, y?: number, options?: IPdfSvgDrawSettings): PdfPageArea
```

Draws a SVG image with the given options.

If x and y are not defined, then @see:x and @see:y are used instead.

The method uses the values of the width and height attributes of the outermost svg element to determine the scale factor according to the options.width and options.height properties. If any of these attributes are omitted then scaling is not performed and the image will be rendered in its original size.

Finally, if the image was drawn in the text flow, the method updates @see:y. Hence, any subsequent text or image starts below this point. The increment value is defined by the options.height property or by the outermost svg element's height attribute, which comes first. If none of them is provided then @see:y will stay unchanged.

The method supports a limited set of SVG features and provided primarily for rendering wijmo 5 chart controls.

Parameters

- **url**: **string**
  A string containing the URL to get the SVG image from or the data URI containing a base64 encoded SVG image.
- **x**: **number** OPTIONAL
  The x-coordinate of the point to draw the image at, in points.
- **y**: **number** OPTIONAL
  The y-coordinate of the point to draw the image at, in points.
- **options**: **IPdfSvgDrawSettings** OPTIONAL
  Determines the SVG image drawing options.

Returns
PdfPageArea

drawText

```
drawText(text: string, x?: number, y?: number, options?: IPdfTextDrawSettings): IPdfTextMeasurementInfo
```

Draws a string with the given options and returns the measurement information.

If **options.pen**, **options.brush** or **options.font** are omitted, the current document's pen, brush or font are used (see **setPen**, **setBrush**, and **setFont**).

The string is drawn within the rectangular area for which top-left corner, width and height are defined by the x, y, **options.width** and **options.height** values. If x and y are not provided, the **x** and **y** properties are used instead.

The text is wrapped and clipped automatically within the area. If **options.height** is not provided and the text exceeds the bottom body edge, then a new page will be added to accommodate the text.

Finally, the method updates the value of the **x** and **y** properties. Hence, any subsequent text or image starts below this point (depending on the value of **options.continued**).

The measurement result doesn't reflect the fact that text can be split into multiple pages or columns; the text is treated as a single block.

Parameters

- **text**: **string**
  The text to draw.
- **x**: **number** OPTIONAL
  The X-coordinate of the point to draw the text at, in points.
- **y**: **number** OPTIONAL
  The Y-coordinate of the point to draw the text at, in points.
- **options**: **IPdfTextDrawSettings** OPTIONAL
  Determines the text drawing options.

Returns
`IPdfTextMeasurementInfo`

lineHeight

```
lineHeight(font?: PdfFont): number
```

Gets the line height with a given font.

If font is not specified, then font used in the current document is used.

Parameters

- **font**: **PdfFont** OPTIONAL
  Font to get the line height.

Returns
number

measureText

```
measureText(text: string, font?: PdfFont, options?: IPdfTextMeasurementSettings): IPdfTextMeasurementInfo
```

Measures a text with the given font and text drawing options without rendering it.

If font is not specified, then the font used in the current document is used.

The method uses the same text rendering engine as **drawText**, so it is tied up in the same way to @see:x and the right page margin, if options.width is not provided. The measurement result doesn't reflect the fact that text can be split into multiple pages or columns; the text is treated as a single block.

Parameters

- **text**: **string**
  Text to measure.

- **font**: **PdfFont** OPTIONAL
  Font to be applied on the text.

- **options**: **IPdfTextMeasurementSettings** OPTIONAL
  Determines the text drawing options.

Returns
IPdfTextMeasurementInfo

moveDown

```
moveDown(lines?: number, font?: PdfFont): PdfPageArea
```

Moves down the @see:y by a given number of lines using the given font or, using the font of current document, if not specified.

Parameters

- **lines**: **number** OPTIONAL
  Number of lines to move down.

- **font**: **PdfFont** OPTIONAL
  Font to calculate the line height.

Returns
`PdfPageArea`

moveUp

```
moveUp(lines?: number, font?: PdfFont): PdfPageArea
```

Moves up the @see:y by a given number of lines using the given font or, using the font of current document, if not specified.

Parameters

- **lines**: **number** OPTIONAL
  Number of lines to move up.

- **font**: **PdfFont** OPTIONAL
  Font to calculate the line height.

Returns
`PdfPageArea`

rotate

```
rotate(angle: number, origin?: Point): PdfPageArea
```

Rotates the graphic context clockwise by a specified angle.

Parameters

- **angle**: **number**
  The rotation angle, in degrees.

- **origin**: **Point** OPTIONAL
  The **`Point`** of rotation, in points. If it is not provided, then the top left corner is used.

Returns
`PdfPageArea`

scale

```
scale(xFactor: number, yFactor?: number, origin?: Point): PdfPageArea
```

Scales the graphic context by a specified scaling factor.

The scaling factor value within the range [0, 1] indicates that the size will be decreased. The scaling factor value greater than 1 indicates that the size will be increased.

Parameters

- **xFactor**: **number**
  The factor to scale the X dimension.

- **yFactor**: **number** OPTIONAL
  The factor to scale the Y dimension. If it is not provided, it is assumed to be equal to xFactor.

- **origin**: **Point** OPTIONAL
  The **`Point`** to scale around, in points. If it is not provided, then the top left corner is used.

Returns
`PdfPageArea`

transform

```
transform(a: number, b: number, c: number, d: number, e: number, f: number): PdfPageArea
```

Transforms the graphic context with given six numbers which represents a 3x3 transformation matrix.

A transformation matrix is written as follows:

ab0
cd0
ef 1

Parameters

- **a**: **number**
  Value of the first row and first column.
- **b**: **number**
  Value of the first row and second column.
- **c**: **number**
  Value of the second row and first column.
- **d**: **number**
  Value of the second row and second column.
- **e**: **number**
  Value of the third row and first column.
- **f**: **number**
  Value of the third row and second column.

Returns
`PdfPageArea`

translate

```
translate(x: number, y: number): PdfPageArea
```

Translates the graphic context with a given distance.

Parameters

- **x**: **number**
  The distance to translate along the X-axis, in points.

- **y**: **number**
  The distance to translate along the Y-axis, in points.

Returns
PdfPageArea

PdfPaths Class

File
`wijmo.pdf.js`
Module
wijmo.pdf

Provides methods for creating graphics paths and drawing them or using them for clipping.

Path creation method calls must be finished with the **stroke**, **fill**, **fillAndStroke** or **clip** method. Any document methods which don't apply directly to path creation/ drawing/ clipping (changing a pen, drawing a text, saving the graphics state etc) are prohibited to use until the path is finished. The **lineTo**, **bezierCurveTo** and **quadraticCurveTo** methods should not start the path, they must be preceded with the **moveTo**.

The methods are chainable:

```
doc.paths.moveTo(0, 0).lineTo(100, 100).stroke();
```

This class is not intended to be instantiated in your code.

Constructor

- constructor

Methods

- bezierCurveTo
- circle
- clip
- closePath
- ellipse
- fill
- fillAndStroke
- lineTo
- moveTo
- polygon
- quadraticCurveTo
- rect
- roundedRect
- stroke
- svgPath

Constructor

constructor

```
constructor(doc: PdfDocument, offset: Point): PdfPaths
```

Initializes a new instance of the **PdfPaths** class.

Parameters

- **doc**: **PdfDocument**
  Document.

- **offset**: **Point**
  Offset.

Returns
PdfPaths

Methods

bezierCurveTo

```
bezierCurveTo(cp1x: number, cp1y: number, cp2x: number, cp2y: number, x: number, y: number): PdfPaths
```

Draws a bezier curve from the current point to a new point using the (cp1x, cp1y) and (cp2x, cp2y) as the control points.

The new current point is (x, y).

Parameters

- **cp1x**: **number**
  The X-coordinate of the first control point, in points.
- **cp1y**: **number**
  The Y-coordinate of the first control point, in points.
- **cp2x**: **number**
  The X-coordinate of the second control point, in points.
- **cp2y**: **number**
  The Y-coordinate of the second control point, in points.
- **x**: **number**
  The X-coordinate of the new point, in points.
- **y**: **number**
  The Y-coordinate of the new point, in points.

Returns
`PdfPaths`

circle

```
circle(x: number, y: number, radius: number): PdfPaths
```

Draws a circle.

Parameters

- **x**: **number**
  The X-coordinate of the center of the circle, in points.
- **y**: **number**
  The Y-coordinate of the center of the circle, in points.
- **radius**: **number**
  The radius of the circle, in points.

Returns
PdfPaths

clip

```
clip(rule?: PdfFillRule): PdfPaths
```

Creates a clipping path used to limit the regions of the page affected by painting operators.

Parameters

- **rule**: **PdfFillRule** OPTIONAL
  The fill rule to use.

Returns
PdfPaths

closePath

```
closePath(): PdfPaths
```

Closes the current path and draws a line from the current point to the initial point of the current path.

Returns
PdfPaths

ellipse

```
ellipse(x: number, y: number, radiusX: number, radiusY?: number): PdfPaths
```

Draws an ellipse.

Parameters

- **x**: **number**
  The X-coordinate of the center of the ellipse, in points.
- **y**: **number**
  The Y-coordinate of the center of the ellipse, in points.
- **radiusX**: **number**
  The radius of the ellipse along the X-axis, in points.
- **radiusY**: **number** OPTIONAL
  The radius of the ellipse along the Y-axis, in points. If it is not provided, then it is assumed to be equal to radiusX.

Returns
`PdfPaths`

fill

```
fill(brushOrColor?: any, rule?: PdfFillRule): PdfPaths
```

Fills the path with the specified brush and rule. If brush is not specified, then the default document brush will be used (see the **`setBrush`** method).

The brushOrColor argument can accept the following values:

- A **PdfBrush** object.
- A **Color** object or any string acceptable by the **fromString** method. In this case, the **PdfBrush** object with the specified color will be created internally.

Parameters

- **brushOrColor**: **any** OPTIONAL
  The brush or color to use.
- **rule**: **PdfFillRule** OPTIONAL
  The fill rule to use.

Returns
`PdfPaths`

fillAndStroke

```
fillAndStroke(brushOrColor?: any, penOrColor?: any, rule?: PdfFillRule): PdfPaths
```

Fills and strokes the path with the specified brush, pen and rule. If brush and pen is not specified, then the default document brush and pen will be used (See the **setBrush**, **setPen** methods).

The brushOrColor argument can accept the following values:

- A **PdfBrush** object.
- A **Color** object or any string acceptable by the **fromString** method. In this case, the **PdfBrush** object with the specified color will be created internally.

The penOrColor argument can accept the following values:

- A **PdfPen** object.
- A **Color** object or any string acceptable by the **fromString** method. In this case, the **PdfPen** object with the specified color will be created internally.

Parameters

- **brushOrColor**: **any** OPTIONAL
  The brush or color to use.
- **penOrColor**: **any** OPTIONAL
  The pen or color to use.
- **rule**: **PdfFillRule** OPTIONAL
  The fill rule to use.

Returns
`PdfPaths`

lineTo

```
lineTo(x: number, y: number): PdfPaths
```

Draws a line from the current point to a new point.

The new current point is (x, y).

Parameters

- **x**: **number**
  The X-coordinate of the new point, in points.
- **y**: **number**
  The Y-coordinate of the new point, in points.

Returns
PdfPaths

moveTo

```
moveTo(x: number, y: number): PdfPaths
```

Sets a new current point.

Parameters

- **x**: **number**
  The X-coordinate of the new point, in points.
- **y**: **number**
  The Y-coordinate of the new point, in points.

Returns
PdfPaths

polygon

```
polygon(points: number[][]): PdfPaths
```

Draws a polygon using a given points array.

Parameters

- **points**: **number**[][]
  An array of two-elements arrays [x, y] specifying the X and Y coordinates of the point, in points.

Returns
PdfPaths

quadraticCurveTo

```
quadraticCurveTo(cpx: number, cpy: number, x: number, y: number): PdfPaths
```

Draws a quadratic curve from the current point to a new point using the current point and (cpx, cpy) as the control points.

The new current point is (x, y).

Parameters

- **cpx**: **number**
  The X-coordinate of the control point, in points.
- **cpy**: **number**
  The Y-coordinate of the control point, in points.
- **x**: **number**
  The X-coordinate of the new point, in points.
- **y**: **number**
  The Y-coordinate of the new point, in points.

Returns
PdfPaths

rect

```
rect(x: number, y: number, width: number, height: number): PdfPaths
```

Draws a rectangle.

Parameters

- **x**: **number**
  The X-coordinate of the topleft corner of the rectangle, in points.
- **y**: **number**
  The Y-coordinate of the topleft corner of the rectangle, in points.
- **width**: **number**
  The width of the rectangle, in points.
- **height**: **number**
  The width of the rectangle, in points.

Returns
`PdfPaths`

roundedRect

```
roundedRect(x: number, y: number, width: number, height: number, cornerRadius?: number): PdfPaths
```

Draws a rounded rectangle.

Parameters

- **x**: **number**
  The X-coordinate of the upper-left corner of the rectangle, in points.
- **y**: **number**
  The Y-coordinate of the upper-left corner of the rectangle, in points.
- **width**: **number**
  The width of the rectangle, in points.
- **height**: **number**
  The width of the rectangle, in points.
- **cornerRadius**: **number** OPTIONAL
  The corner radius of the rectangle, in points. The default value is 0.

Returns
`PdfPaths`

stroke

```
stroke(penOrColor?: any): PdfPaths
```

Strokes the path with the specified pen. If pen is not specified, then the default document pen will be used (See the **`setPen`** method).

The penOrColor argument can accept the following values:

- A **PdfPen** object.
- A **Color** object or any string acceptable by the **fromString** method. In this case, the **PdfPen** object with the specified color will be created internally.

Parameters

- **penOrColor**: **any** OPTIONAL
  The pen or color to use.

Returns
`PdfPaths`

svgPath

```
svgPath(path: string): PdfPaths
```

Draws a SVG 1.1 path.

Parameters

- **path**: **string**
  The SVG path to draw.

Returns
PdfPaths

PdfPen Class

File
`wijmo.pdf.js`
Module
wijmo.pdf

Determines an object used to stroke paths and text.

Constructor

- constructor

Properties

- brush
- cap
- color
- dashPattern
- join
- miterLimit
- width

Methods

- clone
- equals

Constructor

constructor

```
constructor(colorOrBrushOrOptions?: any, width?: number, dashPattern?: PdfDashPattern, cap?: PdfLineCapStyle, join?: PdfLineJoinStyle, miterLimit?: number): PdfPen
```

Initializes a new instance of the **PdfPen** class with the specified color or brush or JavaScript object.

The first argument can accept the following values:

- **Color** object or any string acceptable by the **fromString** method.
- **PdfBrush** object.
- JavaScript object containing initialization properties (all other arguments are ignored).

Parameters

- **colorOrBrushOrOptions**: **any** OPTIONAL
  The color or brush or JavaScript object to use.
- **width**: **number** OPTIONAL
  The width to use.
- **dashPattern**: **PdfDashPattern** OPTIONAL
  The dash pattern to use.
- **cap**: **PdfLineCapStyle** OPTIONAL
  The line cap style to use.
- **join**: **PdfLineJoinStyle** OPTIONAL
  The line join style to use.
- **miterLimit**: **number** OPTIONAL
  The miter limit to use.

Returns
PdfPen

Properties

brush

Gets or sets the brush used to stroke paths. Takes precedence over the **color** property, if defined.

Type
PdfBrush

cap

Gets or sets the shape that shall be used at the open ends of a stroked path. The default value is **Butt**.

Type
`PdfLineCapStyle`

color

Gets or sets the color used to stroke paths. The default color is black.

Type
`Color`

dashPattern

Gets the dash pattern used to stroke paths. The default value is a solid line.

Type
`PdfDashPattern`

join

Gets or sets the shape to be used at the corners of a stroked path. The default value is **Miter**.

Type
`PdfLineJoinStyle`

miterLimit

Determines the maximum value of the miter length to the line width ratio, when the line join is converted from miter to bevel. The default value is 10.

Type
number

width

Gets or sets the line width used to stroke paths, in points. The default width is 1.

Type
number

Methods

clone

```
clone(): PdfPen
```

Creates a copy of this **PdfPen**.

Returns
PdfPen

equals

```
equals(value: PdfPen): boolean
```

Determines whether the specified **PdfPen** instance is equal to the current one.

Parameters

- **value**: **PdfPen**

  PdfPen to compare.

Returns
boolean

PdfRadialGradientBrush Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
PdfGradientBrush

Represents a brush used to fill an area with a radial gradient.

Constructor

- constructor

Properties

- opacity
- r1
- r2
- stops
- x1
- x2
- y1
- y2

Methods

- clone
- equals

Constructor

constructor

```
constructor(x1: number, y1: number, r1: number, x2: number, y2: number, r2: number, stops: PdfGradientStop[], opacity?: number): PdfRadialGradientBrush
```

Initializes a new instance of the **`PdfRadialGradientBrush`** class.

Parameters

- **x1**: **number**
  The X-coordinate of the inner circle's center of the radial gradient.
- **y1**: **number**
  The Y-coordinate of the inner circle's center of the radial gradient.
- **r1**: **number**
  The radius of the inner circle of the radial gradient.
- **x2**: **number**
  The X-coordinate of the outer circle's center of the radial gradient.
- **y2**: **number**
  The Y-coordinate of the outer circle's center of the radial gradient.
- **r2**: **number**
  The radius of the outer circle of the radial gradient.
- **stops**: **PdfGradientStop**[]
  The **`PdfGradientStop`** array to set on this brush.
- **opacity**: **number** OPTIONAL
  The opacity of this brush.

Returns
PdfRadialGradientBrush

Properties

opacity

Gets or sets the opacity of the brush. The value must be in range [0, 1], where 0 indicates that the brush is completely transparent and 1 indicates that the brush is completely opaque. The default value is 1.

Inherited From
`PdfGradientBrush`
Type
number

r1

Gets or sets the radius of the inner circle that represents the starting point of the radial gradient, in page area coordinates, in points.

Type
number

r2

Gets or sets the radius of the outer circle that represents the ending point of the radial gradient, in page area coordinates, in points.

Type
number

stops

Gets or sets an array of **`PdfGradientStop`** objects representing a color, offset and opacity within the brush's gradient axis. The default value is an empty array.

Inherited From
`PdfGradientBrush`
Type
`PdfGradientStop`[]

x1

Gets or sets the X-coordinate of the inner circle's center that represents the starting point of the radial gradient, in page area coordinates, in points.

Type
number

x2

Gets or sets the X-coordinate of the outer circle's center that represents the ending point of the radial gradient, in page area coordinates, in points.

Type
number

y1

Gets or sets the Y-coordinate of the inner circle's center that represents the starting point of the radial gradient, in page area coordinates, in points.

Type
number

y2

Gets or sets the Y-coordinate of the outer circle's center that represents the ending point of the radial gradient, in page area coordinates, in points.

Type
number

Methods

clone

```
clone(): PdfRadialGradientBrush
```

Creates a copy of this **`PdfRadialGradientBrush`**.

Returns
`PdfRadialGradientBrush`

equals

```
equals(value: PdfRadialGradientBrush): boolean
```

Determines whether the specified **PdfRadialGradientBrush** instance is equal to the current one.

Parameters

- **value**: **PdfRadialGradientBrush**
  PdfRadialGradientBrush to compare.

Returns
boolean

PdfRunningTitle Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
PdfPageArea

Represents a running title of the page, like header and footer.

This class is not intended to be instantiated in your code.

Constructor

- constructor

Properties

- declarative
- height
- lineGap
- paths
- width
- x
- y

Methods

- drawImage
- drawSvg
- drawText
- lineHeight
- measureText
- moveDown
- moveUp
- rotate
- scale
- transform
- translate

Constructor

constructor

```
constructor(options?: any): PdfRunningTitle
```

Initializes a new instance of the **PdfRunningTitle** class.

Parameters

- **options**: **any** OPTIONAL
  An optional object containing initialization settings.

Returns
PdfRunningTitle

Properties

declarative

Gets or sets an object that provides the ability to setup the running title content declaratively.

Type
PdfRunningTitleDeclarativeContent

height

Gets or sets the height of the running title, in points. To hide the running title, set this property to 0. Changing this property has no effect on previous drawings; they will not be resized or clipped.

The default value is 24.

Type
number

lineGap

Gets or sets the spacing between each line of text, in points.

The default value is 0.

Inherited From
PdfPageArea
Type
number

paths

Gets an object that provides ability to draw paths.

Inherited From
`PdfPageArea`
Type
`PdfPaths`

width

Gets the width of the area, in points.

Inherited From
`PdfPageArea`
Type
number

x

Gets or sets the X-coordinate (in points) of the current point in the text flow used to draw a text or an image.

Inherited From
`PdfPageArea`
Type
number

y

Gets or sets the Y-coordinate (in points) of the current point in the text flow used to draw a text or an image.

Inherited From
`PdfPageArea`
Type
number

Methods

drawImage

```
drawImage(url: string, x?: number, y?: number, options?: IPdfImageDrawSettings): PdfPageArea
```

Draws an image in JPG or PNG format with the given options.

If x and y are not defined, then @see:x and @see:y are used instead.

Finally, if the image was drawn in the text flow, the method updates @see:y. Hence, any subsequent text or image starts below this point.

Parameters

- **url**: **string**
  A string containing the URL to get the image from or the data URI containing a base64 encoded image.
- **x**: **number** OPTIONAL
  The x-coordinate of the point to draw the image at, in points.
- **y**: **number** OPTIONAL
  The y-coordinate of the point to draw the image at, in points.
- **options**: **IPdfImageDrawSettings** OPTIONAL
  Determines the image drawing options.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

drawSvg

```
drawSvg(url: string, x?: number, y?: number, options?: IPdfSvgDrawSettings): PdfPageArea
```

Draws a SVG image with the given options.

If x and y are not defined, then @see:x and @see:y are used instead.

The method uses the values of the width and height attributes of the outermost svg element to determine the scale factor according to the options.width and options.height properties. If any of these attributes are omitted then scaling is not performed and the image will be rendered in its original size.

Finally, if the image was drawn in the text flow, the method updates @see:y. Hence, any subsequent text or image starts below this point. The increment value is defined by the options.height property or by the outermost svg element's height attribute, which comes first. If none of them is provided then @see:y will stay unchanged.

The method supports a limited set of SVG features and provided primarily for rendering wijmo 5 chart controls.

Parameters

- **url**: **string**
  A string containing the URL to get the SVG image from or the data URI containing a base64 encoded SVG image.
- **x**: **number** OPTIONAL
  The x-coordinate of the point to draw the image at, in points.
- **y**: **number** OPTIONAL
  The y-coordinate of the point to draw the image at, in points.
- **options**: **IPdfSvgDrawSettings** OPTIONAL
  Determines the SVG image drawing options.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

drawText

```
drawText(text: string, x?: number, y?: number, options?: IPdfTextDrawSettings): IPdfTextMeasurementInfo
```

Draws a string with the given options and returns the measurement information.

If **options.pen**, **options.brush** or **options.font** are omitted, the current document's pen, brush or font are used (see **setPen**, **setBrush**, and **setFont**).

The string is drawn within the rectangular area for which top-left corner, width and height are defined by the x, y, **options.width** and **options.height** values. If x and y are not provided, the **x** and **y** properties are used instead.

The text is wrapped and clipped automatically within the area. If **options.height** is not provided and the text exceeds the bottom body edge, then a new page will be added to accommodate the text.

Finally, the method updates the value of the **x** and **y** properties. Hence, any subsequent text or image starts below this point (depending on the value of **options.continued**).

The measurement result doesn't reflect the fact that text can be split into multiple pages or columns; the text is treated as a single block.

Parameters

- **text**: **string**
  The text to draw.
- **x**: **number** OPTIONAL
  The X-coordinate of the point to draw the text at, in points.
- **y**: **number** OPTIONAL
  The Y-coordinate of the point to draw the text at, in points.
- **options**: **IPdfTextDrawSettings** OPTIONAL
  Determines the text drawing options.

Inherited From
PdfPageArea
Returns
IPdfTextMeasurementInfo

lineHeight

```
lineHeight(font?: PdfFont): number
```

Gets the line height with a given font.

If font is not specified, then font used in the current document is used.

Parameters

- **font**: **PdfFont** OPTIONAL
  Font to get the line height.

Inherited From
PdfPageArea
Returns
number

measureText

```
measureText(text: string, font?: PdfFont, options?: IPdfTextMeasurementSettings): IPdfTextMeasurementInfo
```

Measures a text with the given font and text drawing options without rendering it.

If font is not specified, then the font used in the current document is used.

The method uses the same text rendering engine as **drawText**, so it is tied up in the same way to @see:x and the right page margin, if options.width is not provided. The measurement result doesn't reflect the fact that text can be split into multiple pages or columns; the text is treated as a single block.

Parameters

- **text**: **string**
  Text to measure.

- **font**: **PdfFont** OPTIONAL
  Font to be applied on the text.

- **options**: **IPdfTextMeasurementSettings** OPTIONAL
  Determines the text drawing options.

Inherited From
PdfPageArea
Returns
IPdfTextMeasurementInfo

moveDown

```
moveDown(lines?: number, font?: PdfFont): PdfPageArea
```

Moves down the @see:y by a given number of lines using the given font or, using the font of current document, if not specified.

Parameters

- **lines**: **number** OPTIONAL
  Number of lines to move down.

- **font**: **PdfFont** OPTIONAL
  Font to calculate the line height.

Inherited From
PdfPageArea
Returns
PdfPageArea

moveUp

```
moveUp(lines?: number, font?: PdfFont): PdfPageArea
```

Moves up the @see:y by a given number of lines using the given font or, using the font of current document, if not specified.

Parameters

- **lines**: **number** OPTIONAL
  Number of lines to move up.

- **font**: **PdfFont** OPTIONAL
  Font to calculate the line height.

Inherited From
PdfPageArea
Returns
PdfPageArea

rotate

```
rotate(angle: number, origin?: Point): PdfPageArea
```

Rotates the graphic context clockwise by a specified angle.

Parameters

- **angle**: **number**
  The rotation angle, in degrees.

- **origin**: **Point** OPTIONAL
  The **Point** of rotation, in points. If it is not provided, then the top left corner is used.

Inherited From
PdfPageArea
Returns
PdfPageArea

scale

```
scale(xFactor: number, yFactor?: number, origin?: Point): PdfPageArea
```

Scales the graphic context by a specified scaling factor.

The scaling factor value within the range [0, 1] indicates that the size will be decreased. The scaling factor value greater than 1 indicates that the size will be increased.

Parameters

- **xFactor**: **number**
  The factor to scale the X dimension.

- **yFactor**: **number** OPTIONAL
  The factor to scale the Y dimension. If it is not provided, it is assumed to be equal to xFactor.

- **origin**: **Point** OPTIONAL
  The **Point** to scale around, in points. If it is not provided, then the top left corner is used.

Inherited From
PdfPageArea
Returns
PdfPageArea

transform

```
transform(a: number, b: number, c: number, d: number, e: number, f: number): PdfPageArea
```

Transforms the graphic context with given six numbers which represents a 3x3 transformation matrix.

A transformation matrix is written as follows:

ab0
cd0
ef 1

Parameters

- **a**: **number**
  Value of the first row and first column.
- **b**: **number**
  Value of the first row and second column.
- **c**: **number**
  Value of the second row and first column.
- **d**: **number**
  Value of the second row and second column.
- **e**: **number**
  Value of the third row and first column.
- **f**: **number**
  Value of the third row and second column.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

translate

```
translate(x: number, y: number): PdfPageArea
```

Translates the graphic context with a given distance.

Parameters

- **x**: **number**
  The distance to translate along the X-axis, in points.

- **y**: **number**
  The distance to translate along the Y-axis, in points.

Inherited From
`PdfPageArea`
Returns
`PdfPageArea`

PdfRunningTitleDeclarativeContent Class

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the declarative content of the running title.

Constructor

- constructor

Properties

- brush
- font
- text

Methods

- clone
- equals

Constructor

constructor

```
constructor(text?: string, font?: PdfFont, brushOrColor?: any): PdfRunningTitleDeclarativeContent
```

Initializes a new instance of the **`PdfRunningTitleDeclarativeContent`** class.

Parameters

- **text**: **string** OPTIONAL
  The text of the running title.
- **font**: **PdfFont** OPTIONAL
  Font of the text.
- **brushOrColor**: **any** OPTIONAL
  The **`PdfBrush`** or **`Color`** or any string acceptable by the **`fromString`** method used to fill the text.

Returns
PdfRunningTitleDeclarativeContent

Properties

brush

Gets or sets the brush used to fill the **`text`**.

Type
`PdfBrush`

font

Gets or sets the font of the **`text`**.

Type
`PdfFont`

text

Gets or sets the text of the running title.

May contain up to 3 tabular characters ('\t') which are used for separating the text into the parts that will be aligned within the page area using left, center and right alignment. Two kinds of macros are supported, '&[Page]' and '&[Pages]'. The former one designates the current page index while the latter one designates the page count.

For example, for the first page of a document having ten pages, the following string:

```
'&[Page]\\&[Pages]\theader\t&[Page]\\&[Pages]'
```

will be translated to:

```
'1\10 header 1\10'
```

Type
string

Methods

clone

```
clone(): PdfRunningTitleDeclarativeContent
```

Creates a copy of this **PdfRunningTitleDeclarativeContent**.

Returns
PdfRunningTitleDeclarativeContent

equals

```
equals(value: PdfRunningTitleDeclarativeContent): boolean
```

Determines whether the specified **PdfRunningTitleDeclarativeContent** instance is equal to the current one.

Parameters

- **value**: **PdfRunningTitleDeclarativeContent**
  PdfRunningTitleDeclarativeContent to compare.

Returns
boolean

PdfSolidBrush Class

File
`wijmo.pdf.js`
Module
wijmo.pdf
Base Class
PdfBrush

Represents a brush used to fill an area with a color.

Constructor

- constructor

Properties

- color

Methods

- clone
- equals

Constructor

constructor

```
constructor(color?: any): PdfSolidBrush
```

Initializes a new instance of the **PdfSolidBrush** class.

Parameters

- **color**: **any** OPTIONAL
  The color of this brush. A **Color** object or any string acceptable by the **fromString** method.

Returns
PdfSolidBrush

Properties

color

Gets or sets the color of the brush. The default color is black.

Type
`Color`

Methods

clone

```
clone(): PdfSolidBrush
```

Creates a copy of this **PdfSolidBrush**.

Returns
`PdfSolidBrush`

equals

```
equals(value: PdfSolidBrush): boolean
```

Determines whether the specified **PdfSolidBrush** instance is equal to the current one.

Parameters

- **value**: **PdfSolidBrush**
  PdfSolidBrush to compare.

Returns
boolean

IPdfBufferedPageRange Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents a range of buffered pages returned by **bufferedPageRange** method.

Properties

- count
- start

Properties

- **count**

Determines the count of buffered pages.

Type
number

- **start**

Determines the zero-based index of the first buffered page.

Type
number

IPdfDocumentInfo Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the document information used by **info** property.

Properties

- author
- creationDate
- keywords
- modDate
- subject
- title

Properties

author

Determines the name of the person who created the document.

Type
string

creationDate

Determines the date and time the document was created on.

Type
Date

keywords

Determines the keywords associated with the document.

Type
string

modDate

Determines the date and time when the document was last modified.

Type
Date

subject

Determines the subject of the document.

Type
string

title

Determines the title of the document.

Type
string

IPdfFontAttributes Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the font attributes.

Properties

- cursive
- fantasy
- monospace
- sansSerif
- serif

Properties

cursive

Glyphs have finishing strokes, flared or tapering ends, or have actual serifed endings.

Type
boolean

fantasy

Fantasy fonts are primarily decorative fonts that contain playful representations of characters.

Type
boolean

monospace

All glyphs have the same width.

Type
boolean

sansSerif

Glyphs have stroke endings that are plain.

Type
boolean

serif

Glyphs have finishing strokes, flared or tapering ends, or have actual serifed endings.

Type
boolean

IPdfFontFile Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf
Implements
IPdfFontAttributes

Represents the settings of the font to register by **registerFont** and **registerFontAsync** methods.

Properties

- family
- name
- source
- style
- weight

Properties

family

An optional parameter determining the TrueType Collection or Datafork TrueType font family.

Type
string

name

The name of the font to use.

Type
string

source

An ArrayBuffer containing binary data or URL to load the font from. Following font formats are supported: TrueType (.ttf), TrueType Collection (.ttc), Datafork TrueType (.dfont).

Type
any

style

The style of the font. One of the following values: 'normal', 'italic', 'oblique'.

Type
string

weight

The weight of the font. One of the following values: 'normal', 'bold', '100', '200', '300', '400', '500', '600', '700', '800', '900'.

Type
string

IPdfImageDrawSettings Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the image drawing settings used by **drawImage** method.

If neither width nor height options are provided, then the image will be rendered in its original size. If only width is provided, then the image will be scaled proportionally to fit in the provided width. If only height is provided, then the image will be scaled proportionally to fit in the provided height. If both width and height are provided, then image will be stretched to the dimensions depending on the stretchProportionally property.

Properties

- align
- height
- stretchProportionally
- vAlign
- width

Properties

align

Determines the horizontal alignment in case of proportional stretching.

Type
`PdfImageHorizontalAlign`

height

Determines the height of the image, in points.

Type
number

stretchProportionally

Indicates whether an image will be stretched proportionally or not, if both width and height options are provided.

Type
boolean

vAlign

Determines the vertical alignment in case of proportional stretching.

Type
`PdfImageVerticalAlign`

width

Determines the width of the image, in points.

Type
number

IPdfPageMargins Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the page margins.

Properties

- bottom
- left
- right
- top

Properties

bottom

Determines the bottom margin, in points.

Type
number

left

Determines the left margin, in points.

Type
number

right

Determines the right margin, in points.

Type
number

top

Determines the top margin, in points.

Type
number

IPdfPageSettings Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the page settings.

Properties

- layout
- margins
- size

Properties

layout

Determines the layout of the page.

Type
`PdfPageOrientation`

margins

Determines the margins of the page.

Type
`IPdfPageMargins`

size

Determines the dimensions of the page. The following values are supported:

- **PdfPageSize**: predefined sizes.
- **Size**: custom sizes.

Type
any

IPdfSvgDrawSettings Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf
Implements
IPdfImageDrawSettings

Represents the settings used by **drawSvg** method to draw a SVG image.

Properties

- urlResolver

Properties

- **urlResolver**

Determines a callback function used to convert a relative URL to a URL that is correct for the current request path. The function gets passed the relative URL as its argument and should return the resolved URL.

Type
Function

IPdfTextDrawSettings Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf
Implements
IPdfTextSettings

Represents the settings used by **drawText** method to draw a text with the specified **PdfPen** and **PdfBrush**.

Properties

- brush
- font
- pen

Properties

brush

Determines the brush to fill the text. If not specified, the default document brush will be used (**`setBrush`** method).

Type
any

font

Determines the font to use. If not specified, the default document font will be used (**`setFont`** method).

Type
`PdfFont`

pen

Determines the pen to stroke the text. If not specified, the default document pen will be used (**`setPen`** method).

Type
any

IPdfTextMeasurementInfo Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents the text measurement information returned by **measureText** method.

Properties

- charCount
- size

Properties

charCount

Determines the character count.

Type
number

size

Determines the text size, in points.

Type
Size

IPdfTextMeasurementSettings Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf
Implements
IPdfTextSettings

Represents the settings used by **measureText** method.

Properties

- includeLastLineExternalLeading

Properties

- **includeLastLineExternalLeading**

Determines whether the last line external leading value should be included into the measurements result. The default value is true.

Type
boolean

IPdfTextSettings Interface

File
`wijmo.pdf.js`
Module
wijmo.pdf

Represents text settings used by **drawText** and **measureText** methods.

Properties

- align
- characterSpacing
- columnGap
- columns
- continued
- ellipsis
- fill
- height
- indent
- lineBreak
- lineGap
- link
- paragraphGap
- strike
- stroke
- underline
- width
- wordSpacing

Properties

align

Determines how text is aligned within the drawing area. The default value is **Left**.

Type
`PdfTextHorizontalAlign`

characterSpacing

Determines the spacing between text characters. The default value is 0.

Type
number

columnGap

Determines the spacing between each column, in points. The default value is 18.

Type
number

columns

Determines the number of columns to flow the text into. The default value is 1.

Type
number

continued

Indicates whether subsequent text should be continued right after that or it will be a new paragraph. If true, the text settings will be retained between drawText calls. It means that options argument will be merged with the one taken from the previous drawText call.

The default value is false.

Type
boolean

ellipsis

Determines the character to display at the end of the text when it exceeds the given area.The default value is undefined, that is, ellipsis is not displayed. Set to true to use the default character.

Type
any

fill

Indicates whether the text should be filled or not. The default value is true.

Type
boolean

height

Determines the height of the drawing area in points to which the text should be clipped. The default value is undefined which means that the text area will be limited by bottom edge of the body section. Use Infinity to indicate that the text area has an infinite height.

Type
number

indent

Determines the value of indentaion in each paragraph of text, in points. The default value is 0.

Type
number

lineBreak

Indicates whether line wrapping should be used or not. The property is ignored if **`width`** is defined. The default value is true.

Type
boolean

lineGap

Determines the spacing between lines of text. The default value is 0.

Type
number

link

Determines a URL used to create a link annotation (URI action).

Type
string

paragraphGap

Determines the spacing between paragraphs of text. The default value is 0.

Type
number

strike

Indicates whether the text should be striked out or not. The default value is false.

Type
boolean

stroke

Indicates whether the text should be stroked or not. The default value is false.

Type
boolean

underline

Indicates whether the text should be underlined or not. The default value is false.

Type
boolean

width

Determines the width of the text area in points to which the text should wrap. The default value is undefined which means that the text area will be limited by right margin of the page. Use Infinity to indicate that the text area has an infinite width. If defined, forces the **`lineBreak`** property to be enabled.

Type
number

wordSpacing

Determines the spacing between words in the text. The default value is 0.

Type
number

PdfFillRule Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies a rule that determines if a point falls inside the enclosed path.

Members

Name	Value	Description
NonZero	0	Non-zero rule.
EvenOdd	1	Even-odd rule.

PdfImageHorizontalAlign Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the horizontal alignment of the image.

Members

Name	Value	Description
Left	0	Aligns the image to the left edge of the drawing area.
Center	1	Aligns the image in the middle of the drawing area.
Right	2	Aligns the image to the right edge of the drawing area.

PdfImageVerticalAlign Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the vertical alignment of the image.

Members

Name	Value	Description
Top	0	Aligns the image to the top edge of the drawing area.
Center	1	Aligns the image in the middle of the drawing area.
Bottom	2	Aligns the image to the bottom edge of the drawing area.

PdfLineCapStyle Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the shape that shall be used at the ends of open subpaths (and dashes, if any) when they are stroked.

Members

Name	Value	Description
Butt	0	The stroke is squared off at the endpoint of the path.
Round	1	A semicircular arc with a diameter equal to the line width is drawn around the endpoint and is filled in.
Square	2	The stroke continues beyond the endpoint of the path for a distance equal to the half of the line width and is squared off.

PdfLineJoinStyle Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the shape to be used at the corners of paths that are stroked.

Members

Name	Value	Description
Miter	0	The outer edges of the strokes for the two segments are extended until they meet at an angle.
Round	1	An arc of a circle with a diameter equal to the line width is drawn around the point where the two segments meet.
Bevel	2	The two segments are finished with butt caps and the resulting notch beyond the ends of the segments is filled with a triangle.

PdfPageOrientation Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the page orientation.

Members

Name	Value	Description
Portrait	0	Portrait orientation.
Landscape	1	Landscape orientation.

PdfPageSize Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the page size, in points.

Members

Name	Value	Description
A0	0	Represents the A0 page size.
A1	1	Represents the A1 page size.
A2	2	Represents the A2 page size.
A3	3	Represents the A3 page size.
A4	4	Represents the A4 page size.
A5	5	Represents the A5 page size.
A6	6	Represents the A6 page size.
A7	7	Represents the A7 page size.
A8	8	Represents the A8 page size.
A9	9	Represents the A9 page size.
A10	10	Represents the A10 page size.
B0	11	Represents the B0 page size.
B1	12	Represents the B1 page size.
B2	13	Represents the B2 page size.
B3	14	Represents the B3 page size.
B4	15	Represents the B4 page size.
B5	16	Represents the B5 page size.
B6	17	Represents the B6 page size.
B7	18	Represents the B7 page size.
B8	19	Represents the B8 page size.
B9	20	Represents the B9 page size.
B10	21	Represents the B10 page size.
C0	22	Represents the C0 page size.
C1	23	Represents the C1 page size.
C2	24	Represents the C2 page size.
C3	25	Represents the C3 page size.
C4	26	Represents the C4 page size.
C5	27	Represents the C5 page size.
C6	28	Represents the C6 page size.
C7	29	Represents the C7 page size.
C8	30	Represents the C8 page size.
C9	31	Represents the C9 page size.
C10	32	Represents the C10 page size.
RA0	33	Represents the RA0 page size.
RA1	34	Represents the RA1 page size.
RA2	35	Represents the RA2 page size.
RA3	36	Represents the RA3 page size.
RA4	37	Represents the RA4 page size.
SRA0	38	Represents the SRA0 page size.

Name	Value	Description
SRA1	39	Represents the SRA1 page size.
SRA2	40	Represents the SRA2 page size.
SRA3	41	Represents the SRA3 page size.
SRA4	42	Represents the SRA4 page size.
Executive	43	Represents the executive page size.
Folio	44	Represents the folio page size.
Legal	45	Represents the legal page size.
Letter	46	Represents the letter page size.
Tabloid	47	Represents the tabloid page size.

PdfTextHorizontalAlign Enum

File
`wijmo.pdf.js`
Module
wijmo.pdf

Specifies the horizontal alignment of text content.

Members

Name	Value	Description
Left	0	Text is aligned to the left.
Center	1	Text is centered.
Right	2	Text is aligned to the right.
Justify	3	Text is justified.

wijmo.grid.pdf Module

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Defines the **FlexGridPdfConverter** class used to export the **FlexGrid** to PDF.

Classes

- FlexGridPdfConverter
- PdfFormatItemEventArgs

Interfaces

- ICellStyle
- IFlexGridDrawSettings
- IFlexGridExportSettings
- IFlexGridStyle

Enums

- ExportMode
- ScaleMode

FlexGridPdfConverter Class

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Provides a functionality to export the **FlexGrid** to PDF.

Methods

- draw
- drawToPosition
- export

Methods

STATIC draw

```
draw(flex: FlexGrid, doc: PdfDocument, width?: number, height?: number, settings?: IFlexGridDrawSettings): void
```

Draws the **`FlexGrid`** to an existing **`PdfDocument`** at the (0, @wijmo.pdf.PdfDocument.y) coordinates.

If width is not specified, then grid will be rendered in actual size, breaking into pages as needed. If height is not specified, then grid will be scaled to fit the width, breaking into pages vertically as needed. If both, width and height are determined, then grid will be scaled to fit the specified rectangle without any page breaks.

```
var doc = new wijmo.pdf.PdfDocument({
   ended: function (sender, args) {
      wijmo.pdf.saveBlob(args.blob, 'FlexGrid.pdf');
   }
});

wijmo.grid.pdf.FlexGridPdfConverter.draw(grid, doc, null, null, {
   maxPages: 10,
   styles: {
      cellStyle: {
         backgroundColor: '#ffffff',
         borderColor: '#c6c6c6'
      },
      headerCellStyle: {
         backgroundColor: '#eaeaea'
      }
   }
});
```

Parameters

- **flex**: **FlexGrid**
  The **`FlexGrid`** instance to export.
- **doc**: **PdfDocument**
  The **`PdfDocument`** instance to draw in.
- **width**: **number** OPTIONAL
  The width of the drawing area in points.
- **height**: **number** OPTIONAL
  The height of the drawing area in points.
- **settings**: **IFlexGridDrawSettings** OPTIONAL
  The draw settings.

Returns
void

STATIC drawToPosition

```
drawToPosition(flex: FlexGrid, doc: PdfDocument, point: Point, width?: number, height?: number, settings?: IFlexGridDrawSettings): void
```

Draws the **`FlexGrid`** to an existing **`PdfDocument`** instance at the specified coordinates.

If width is not specified, then grid will be rendered in actual size without any page breaks. If height is not specified, then grid will be scaled to fit the width without any page breaks. If both, width and height are determined, then grid will be scaled to fit the specified rectangle without any page breaks.

```
var doc = new wijmo.pdf.PdfDocument({
   ended: function (sender, args) {
      wijmo.pdf.saveBlob(args.blob, 'FlexGrid.pdf');
   }
});

wijmo.grid.pdf.FlexGridPdfConverter.drawToPosition(grid, doc, new wijmo.Point(0, 0), null, null, {
   maxPages: 10,
   styles: {
      cellStyle: {
         backgroundColor: '#ffffff',
         borderColor: '#c6c6c6'
      },
      headerCellStyle: {
         backgroundColor: '#eaeaea'
      }
   }
});
```

Parameters

- **flex**: **FlexGrid**
  The **`FlexGrid`** instance to export.
- **doc**: **PdfDocument**
  The **`PdfDocument`** instance to draw in.
- **point**: **Point**
  The position to draw at, in points.
- **width**: **number** OPTIONAL
  The width of the drawing area in points.
- **height**: **number** OPTIONAL
  The height of the drawing area in points.
- **settings**: **IFlexGridDrawSettings** OPTIONAL
  The draw settings.

Returns

void

STATIC **export**

```
export(flex: FlexGrid, fileName: string, settings?: IFlexGridExportSettings): void
```

Exports the **FlexGrid** to PDF.

```
wijmo.grid.pdf.FlexGridPdfConverter.export(grid, 'FlexGrid.pdf', {
   scaleMode: wijmo.grid.pdf.ScaleMode.PageWidth,
   maxPages: 10,
   styles: {
      cellStyle: {
         backgroundColor: '#ffffff',
         borderColor: '#c6c6c6'
      },
      headerCellStyle: {
         backgroundColor: '#eaeaea'
      }
   },
   documentOptions: {
      info: {
         title: 'Sample'
      }
   }
});
```

Parameters

- **flex**: **FlexGrid**
  The **FlexGrid** instance to export.
- **fileName**: **string**
  Name of the file to export.
- **settings**: **IFlexGridExportSettings** OPTIONAL
  The export settings.

Returns
void

PdfFormatItemEventArgs Class

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf
Base Class
CellRangeEventArgs

Represents arguments of the IFlexGridDrawSettings.formatItem callback.

Constructor

- constructor

Properties

- cancel
- cancelBorders
- canvas
- cell
- clientRect
- col
- contentRect
- data
- empty
- panel
- range
- row
- style
- textTop

Methods

- getFormattedCell

Constructor

constructor

```
constructor(p: GridPanel, rng: CellRange, cell: HTMLElement, canvas: PdfPageArea, clientRect: Rect, contentRect: Rect, textTop: number, style: ICellStyle, getFormattedCell?: Function): PdfFormatItemEventAr
```

Initializes a new instance of the **`PdfFormatItemEventArgs`** class.

Parameters

- **p**: **GridPanel**
  `GridPanel` that contains the range.
- **rng**: **CellRange**
  Range of cells affected by the event.
- **cell**: **HTMLElement**
  Element that represents the grid cell to be rendered.
- **canvas**: **PdfPageArea**
  Canvas to perform the custom painting on.
- **clientRect**: **Rect**
  Object that represents the client rectangle of the grid cell to be rendered in canvas coordinates.
- **contentRect**: **Rect**
  Object that represents the content rectangle of the grid cell to be rendered in canvas coordinates.
- **textTop**: **number**
  Object that represents the top position of the text in canvas coordinates.
- **style**: **ICellStyle**
  Object that represents the style of the grid cell to be rendered.
- **getFormattedCell**: **Function** OPTIONAL
  Callback function that should return the grid cell when the getFormattedCell method is called.

Returns
PdfFormatItemEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

cancelBorders

Gets or sets a value that indicates that default cell borders drawing should be canceled.

Type
boolean

canvas

Gets the canvas to perform the custom painting on.

Type
`PdfPageArea`

cell

Gets a reference to the element that represents the grid cell being rendered. If IFlexGridDrawSettings.customCellContent is set to true then contains reference to the element that represents the formatted grid cell; otherwise, a null value.

Type
HTMLElement

clientRect

Gets the client rectangle of the cell being rendered in canvas coordinates.

Type
`Rect`

col

Gets the column affected by this event.

Inherited From
`CellRangeEventArgs`
Type
number

contentRect

Gets the content rectangle of the cell being rendered in canvas coordinates.

Type
`Rect`

data

Gets or sets the data associated with the event.

Inherited From
`CellRangeEventArgs`
Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

panel

Gets the **GridPanel** affected by this event.

Inherited From
`CellRangeEventArgs`
Type
`GridPanel`

range

Gets the **CellRange** affected by this event.

Inherited From
CellRangeEventArgs
Type
CellRange

row

Gets the row affected by this event.

Inherited From
CellRangeEventArgs
Type
number

style

Gets an object that represents the style of the cell being rendered. If IFlexGridDrawSettings.customCellContent is set to true then the style is inferred from the cell style; othwerwise it contains a combination of the IFlexGridDrawSettings.styles export setting, according to the row type of exported cell.

Type
ICellStyle

textTop

Gets the value that represents the top position of the text of the cell being rendered in canvas coordinates.

Type
number

Methods

getFormattedCell

```
getFormattedCell(): HTMLElement
```

Returns a reference to the element that represents the grid cell being rendered. This method is useful when export of custom formatting is disabled, but you need to export custom content for certain cells.

Returns
HTMLElement

ICellStyle Interface

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Represents the look and feel of a cell.

Properties

- backgroundColor
- borderColor
- font

Properties

backgroundColor

Represents the background color of a cell.

Type
string

borderColor

Represents the border color of a cell.

Type
string

font

Represents the font of a cell.

Type
any

IFlexGridDrawSettings Interface

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Represents the settings used by the **draw** and **drawToPosition** methods.

Properties

- customCellContent
- embeddedFonts
- exportMode
- formatItem
- maxPages
- recalculateStarWidths
- repeatMergedValuesAcrossPages
- styles

Properties

- **customCellContent**

Indicates whether custom cell content and style should be evaluated and exported. If set to true then export logic will retrieve cell content using cell.textContent property, and cell style using getComputedStyle(cell). Default is 'undefined' (i.e. false).

Type
boolean

embeddedFonts

Represents an array of custom fonts that will be embedded into the document.

This sample illustrates how to setup the FlexGridPdfConverter to use two custom fonts, Cuprum-Bold.ttf and Cuprum-Regular.ttf. The first one is applied to the header cells only, while the second one is applied to all the remaining cells.

```
wijmo.grid.pdf.FlexGridPdfConverter.export(flex, fileName, {
   embeddedFonts: [{
      source: 'resources/ttf/Cuprum-Bold.ttf',
      name: 'cuprum',
      style: 'normal',
      weight: 'bold'
   }, {
      source: 'resources/ttf/Cuprum-Regular.ttf',
      name: 'cuprum',
      style: 'normal',
      weight: 'normal'
   }],
   styles: {
      cellStyle: {
         font: {
            family: 'cuprum'
         }
      },
      headerCellStyle: {
         font: {
            weight: 'bold'
         }
      }
   }
});
```

Type
`IPdfFontFile`[]

exportMode

Determines the export mode.

Type
`ExportMode`

formatItem

An optional callback function called for every exported cell that allows to perform transformations of exported cell value and style, or perform a custom drawing.

The function accepts the **`PdfFormatItemEventArgs`** class instance as the first argument.

In case of custom drawing the **`cancel`** property should be set to true to cancel the default cell content drawing, and the **`cancelBorders`** property should be set to true to cancel the default cell borders drawing.

```
wijmo.grid.pdf.FlexGridPdfConverter.export(flex, fileName, {
   formatItem: function(args) {

       // Change the background color of the regular cells of "Country" column.
       if (args.panel.cellType === wijmo.grid.CellType.Cell && args.panel.columns[args.col].binding === "country") {
           args.style.backgroundColor = 'blue';
       }
   }
});
```

Type
Function

maxPages

Determines the maximum number of pages to export.

Type
number

recalculateStarWidths

Indicates whether star-sized columns widths should be recalculated against the PDF page width instead of using the grid's width.

Type
boolean

repeatMergedValuesAcrossPages

Indicates whether merged values should be repeated across pages when the merged range is split on multiple pages.

Type
boolean

styles

Represents the look and feel of an exported **`FlexGrid`**.

Type
`IFlexGridStyle`

IFlexGridExportSettings Interface

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf
Implements
IFlexGridDrawSettings

Represents the settings used by the **export** method.

Properties

- documentOptions
- scaleMode

Properties

documentOptions

Represents the options of the underlying **`PdfDocument`**.

Type
any

scaleMode

Determines the scale mode.

Type
`ScaleMode`

IFlexGridStyle Interface

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Represents the look and feel of the **FlexGrid** being exported.

Properties

Properties

altCellStyle

Represents the cell style applied to odd-numbered rows of the **`FlexGrid`**.

Type
`ICellStyle`

cellStyle

Specifies the cell style applied to cells within a **`FlexGrid`**.

Type
`ICellStyle`

errorCellStyle

Represents the cell style applied to cells of the **`FlexGrid`** that contain validation errors if the **`showErrors`** property is enabled.

Type
`ICellStyle`

footerCellStyle

Represents the cell style applied to column footers of the **FlexGrid**.

Type
ICellStyle

groupCellStyle

Represents the cell style applied to grouped rows of the **FlexGrid**.

Type
ICellStyle

headerCellStyle

Represents the cell style applied to row headers and column headers of the **FlexGrid**.

Type
ICellStyle

ExportMode Enum

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Specifies whether the whole grid or just a section should be rendered.

Members

Name	Value	Description
All	0	Exports all the data from grid.
Selection	1	Exports the current selection only.

ScaleMode Enum

File
`wijmo.grid.pdf.js`
Module
wijmo.grid.pdf

Specifies how the grid content should be scaled to fit the page.

Members

Name	Value	Description
ActualSize	0	Render the grid in actual size, breaking into pages as needed.
PageWidth	1	Scale the grid, so that it fits the page width.
SinglePage	2	Scale the grid, so that it fits on a single page.

wijmo.nav Module

File
`wijmo.nav.js`
Module
wijmo.nav

Defines navigation controls including the **TreeView** and associated classes.

Classes

- FormatNodeEventArgs
- TreeNode
- TreeNodeDragDropEventArgs
- TreeNodeEventArgs
- TreeView

Enums

- DropPosition

FormatNodeEventArgs Class

File
`wijmo.nav.js`
Module
wijmo.nav
Base Class
EventArgs

Provides arguments for the **formatItem** event.

Constructor

- constructor

Properties

- dataItem
- element
- empty
- level

Constructor

constructor

```
constructor(dataItem: any, element: HTMLElement, level: number): FormatNodeEventArgs
```

Initializes a new instance of the **FormatNodeEventArgs** class.

Parameters

- **dataItem**: **any**
  Data item represented by the node.
- **element**: **HTMLElement**
  Element that represents the node being formatted.
- **level**: **number**
  The outline level of the node being formatted.

Returns
FormatNodeEventArgs

Properties

dataItem

Gets the data item being formatted.

Type
any

element

Gets a reference to the element that represents the node being formatted.

Type
HTMLElement

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

level

Gets the outline level of the node being formatted.

Type
number

TreeNode Class

File
`wijmo.nav.js`
Module
wijmo.nav

Class that represents a node in a **TreeView**.

Constructor

- constructor

Properties

- checkBox
- dataItem
- element
- hasChildren
- hasPendingChildren
- index
- isChecked
- isCollapsed
- isDisabled
- level
- nodes
- parentNode
- treeView

Methods

- ensureVisible
- equals
- move
- next
- nextSibling
- previous
- previousSibling
- select
- setChecked
- setCollapsed

Constructor

constructor

```
constructor(treeView: TreeView, nodeElement: HTMLElement): TreeNode
```

Initializes a new instance of a **TreeNode**.

Parameters

- **treeView**: **TreeView**
  TreeView that contains the node.

- **nodeElement**: **HTMLElement**
  HTML element that represents the node on the **TreeView**.

Returns
TreeNode

Properties

checkBox

Gets the HTMLInputElement that represents the checkbox associated with this node.

Type
HTMLInputElement

dataItem

Gets the data item that this node represents.

Type
any

element

Gets the HTML element that represents this node on the **TreeView**.

Type
any

hasChildren

Gets a value that indicates whether this node has child nodes.

Type
boolean

hasPendingChildren

Gets a value that indicates whether this node has pending child nodes that will be lazy-loaded when the node is expanded.

Type
boolean

index

Gets this node's index within the parent's node collection.

Type
number

isChecked

Gets or sets a value that determines whether this node is checked.

When the value of this property changes, child and ancestor nodes are automatically updated, and the parent **`TreeView`**'s **`checkedItemsChanged`** event is raised.

Type
boolean

isCollapsed

Gets or sets a value that determines whether this node is expanded or collapsed.

Type
boolean

isDisabled

Gets or sets a value that determines whether this node is disabled.

Disabled nodes cannot get mouse or keyboard events.

Type
boolean

level

Gets this node's level.

Top-level nodes have level zero.

Type
number

nodes

Gets an array containing this node's child nodes.

This property returns null if the node has no children.

Type
`TreeNode`[]

parentNode

Gets this node's parent node.

This property returns null for top-level nodes.

Type
`TreeNode`

treeView

Gets a reference to the **`TreeView`** that contains this node.

Type
`TreeView`

Methods

ensureVisible

```
ensureVisible(): void
```

Ensures that a node is visible by expanding any collapsed ancestors and scrolling the element into view.

Returns
void

equals

```
equals(node: TreeNode): boolean
```

Checks whether this node refers to the same element as another node.

Parameters

- **node**: **TreeNode**
  @TreeNode to compare with this one.

Returns
boolean

move

```
move(refNode: TreeNode, position: DropPosition): boolean
```

Moves a **`TreeNode`** to a new position on the **`TreeView`**.

Parameters

- **refNode**: **TreeNode**
  Reference **`TreeNode`** that defines the location where the node will be moved.

- **position**: **DropPosition**
  Whether to move the node before, after, or into the reference node.

Returns
boolean

next

```
next(visible?: boolean, enabled?: boolean): TreeNode
```

Gets a reference to the next node in the view.

Parameters

- **visible**: **boolean** OPTIONAL
  Whether to return only visible nodes (whose ancestors are not collapsed).

- **enabled**: **boolean** OPTIONAL
  Whether to return only enabled nodes (whose ancestors are not disabled).

Returns
TreeNode

nextSibling

```
nextSibling(): TreeNode
```

Gets a reference to the next sibling node in the view.

Returns
TreeNode

previous

```
previous(visible?: boolean, enabled?: boolean): TreeNode
```

Gets a reference to the previous node in the view.

Parameters

- **visible**: **boolean** OPTIONAL
  Whether to return only visible nodes (whose ancestors are not collapsed).

- **enabled**: **boolean** OPTIONAL
  Whether to return only enabled nodes (whose ancestors are not disabled).

Returns
TreeNode

previousSibling

```
previousSibling(): TreeNode
```

Gets a reference to the next sibling node in the view.

Returns
TreeNode

select

```
select(): void
```

Selects this node.

Returns
void

setChecked

```
setChecked(checked: boolean, updateParent?: boolean): void
```

Sets the checked state of this node and its children.

Parameters

- **checked**: **boolean**
  Whether to check or uncheck the node and its children.
- **updateParent**: **boolean** OPTIONAL
  Whether to update the checked state of this node's ancestor nodes.

Returns
void

setCollapsed

```
setCollapsed(collapsed: boolean, animate?: boolean, collapseSiblings?: boolean): void
```

Sets the collapsed state of the node.

Parameters

- **collapsed**: **boolean**
  Whether to collapse or expand the node.

- **animate**: **boolean** OPTIONAL
  Whether to use animation when applying the new state.

- **collapseSiblings**: **boolean** OPTIONAL
  Whether to collapse sibling nodes when expanding this node.

Returns
void

TreeNodeDragDropEventArgs Class

File
`wijmo.nav.js`
Module
wijmo.nav
Base Class
CancelEventArgs

Provides arguments for **TreeNode** drag-drop events.

Constructor

- constructor

Properties

- cancel
- dragSource
- dropTarget
- empty
- position

Constructor

constructor

```
constructor(dragSource: TreeNode, dropTarget: TreeNode, position: DropPosition): TreeNodeDragDropEventArgs
```

Initializes a new instance of the **TreeNodeEventArgs** class.

Parameters

- **dragSource**: **TreeNode**
  TreeNode being dragged.
- **dropTarget**: **TreeNode**
  TreeNode where the source is being dropped.
- **position**: **DropPosition**
  DropPosition that this event refers to.

Returns
TreeNodeDragDropEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

dragSource

Gets a reference to the **`TreeNode`** being dragged.

Type
`TreeNode`

dropTarget

Gets a reference to the current **`TreeNode`** target.

Type
`TreeNode`

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

position

Gets or sets the **`DropPosition`** value that specifies where the **`TreeNode`** will be dropped.

Type
`DropPosition`

TreeNodeEventArgs Class

File
`wijmo.nav.js`
Module
wijmo.nav
Base Class
CancelEventArgs

Provides arguments for **TreeNode**-related events.

Constructor

- constructor

Properties

- cancel
- empty
- node

Constructor

constructor

```
constructor(node: TreeNode): TreeNodeEventArgs
```

Initializes a new instance of the **TreeNodeEventArgs** class.

Parameters

- **node**: **TreeNode**
  TreeNode that this event refers to.

Returns
TreeNodeEventArgs

Properties

cancel

Gets or sets a value that indicates whether the event should be canceled.

Inherited From
`CancelEventArgs`
Type
boolean

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

node

Gets the **`TreeNode`** that this event refers to.

Type
`TreeNode`

TreeView Class

File
`wijmo.nav.js`
Module
wijmo.nav
Base Class
Control
Derived Classes
WjTreeView

The **TreeView** control displays a hierarchical list of **TreeNode** objects which may contain text, checkboxes, images, or arbitrary HTML content.

A **TreeView** is typically used to display the headings in a document, the entries in an index, the files and directories on a disk, or any other kind of information that might usefully be displayed as a hierarchy.

After creating a **TreeView**, you will typically set the following properties:

1. **itemsSource**: an array that contains the data to be displayed on the tree.
2. **displayMemberPath**: the name of the data item property that contains the text to display on the nodes (defaults to 'header'), and
3. **childItemsPath**: the name of the data item property that contains the node's child items (defaults to 'items').

The example below builds a simple tree and allows you to see the effect of the TreeView's main properties:

Example

Show me (http://jsfiddle.net/Wijmo5/egmg93wc)

Constructor

- constructor

Properties

- allowDragging
- autoCollapse
- checkedItems
- childItemsPath
- controlTemplate
- displayMemberPath
- expandOnClick
- hostElement
- imageMemberPath
- isAnimated
- isContentHtml
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemsSource
- lazyLoadFunction
- nodes
- rightToLeft
- selectedItem
- selectedNode
- selectedPath
- showCheckboxes
- totalItemCount

Methods

- addEventListener
- applyTemplate
- beginUpdate
- checkAllItems
- collapseToLevel
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getControl
- getFirstNode
- getLastNode
- getNode
- getTemplate
- initialize
- invalidate
- invalidateAll
- loadTree
- onCheckedItemsChanged
- onDragEnd
- onDragOver
- onDragStart
- onDrop
- onFormatItem
- onGotFocus
- onIsCheckedChanged
- onIsCheckedChanging
- onIsCollapsedChanged
- onIsCollapsedChanging
- onItemClicked
- onItemsSourceChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onNodeEditEnded
- onNodeEditEnding
- onNodeEditStarted
- onNodeEditStarting
- onSelectedItemChanged
- refresh
- refreshAll
- removeEventListener
- startEditing

Events

- checkedItemsChanged
- dragEnd
- dragOver
- dragStart
- drop
- formatItem

- gotFocus
- isCheckedChanged
- isCheckedChanging
- isCollapsedChanged
- isCollapsedChanging
- itemClicked
- itemsSourceChanged
- loadedItems
- loadingItems
- lostFocus
- nodeEditEnded
- nodeEditEnding
- nodeEditStarted
- nodeEditStarting
- selectedItemChanged

Constructor

constructor

```
constructor(element: any, options?): TreeView
```

Initializes a new instance of the **`TreeView`** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
TreeView

Properties

allowDragging

Gets or sets a value that determines whether users can drag and drop nodes within the **`TreeView`**.

Type
boolean

autoCollapse

Gets or sets a value that determines if sibling nodes should be collapsed when a node is expanded.

This property is set to true by default, because in most cases collapsing nodes that are not in use helps keep the UI clearer.

Type
boolean

checkedItems

Gets an array containing the items that are currently checked.

The array returned includes only items that have no children. This is because checkboxes in parent items are used to check or uncheck the child items.

See also the **`showCheckboxes`** property and the **`checkedItemsChanged`** property.

For example:

```
var treeViewChk = new wijmo.input.TreeView('#gsTreeViewChk', {
   displayMemberPath: 'header',
   childItemsPath: 'items',
   showCheckboxes: true,
   itemsSource: items,
   checkedItemsChanged: function (s, e) {
       var items = s.checkedItems,
           msg = '';
       if (items.length) {
           msg = '<p><b>Selected Items:</b></p><ol>\r\n';
           for (var i = 0; i < items.length; i++) {
               msg += '<li>' + items[i].header + '</li>\r\n';
           }
           msg += '</ol>';
       }
       document.getElementById('gsTreeViewChkStatus').innerHTML = msg;
   }
});
```

Type
any[]

childItemsPath

Gets or sets the name of the property (or properties) that contains the child items for each node.

The default value for this property is the string 'items'.

In most cases, the property that contains the child items is the same for all data items on the tree. In these cases, set the **childItemsPath** to that name.

In some cases, however, items at different levels use different properties to store their child items. For example, you could have a tree with categories, products, and orders. In that case, you would set the **childItemsPath** to an array such as this:

```
// categories have products, products have orders:
tree.childItemsPath = [ 'Products', 'Orders' ];
```

Type
any

STATIC controlTemplate

Gets or sets the template used to instantiate **FlexGrid** controls.

Type
any

displayMemberPath

Gets or sets the name of the property (or properties) to use as the visual representation of the nodes.

The default value for this property is the string 'header'.

In most cases, the property that contains the node text is the same for all data items on the tree. In these cases, set the **displayMemberPath** to that name.

In some cases, however, items at different levels use different properties to represent them. For example, you could have a tree with categories, products, and orders. In that case, you might set the **displayMemberPath** to an array such as this:

```
// categories, products, and orders have different headers:
tree.displayMemberPath = [ 'CategoryName', 'ProductName', 'OrderID' ];
```

Type
any

expandOnClick

Gets or sets a value that determines whether to expand collapsed nodes when the user clicks the node header.

Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imageMemberPath

Gets or sets the name of the property (or properties) to use as a source of images for the nodes.

Type
any

isAnimated

Gets or sets a value that indicates whether to use animations when expanding or collapsing nodes.

Type
boolean

isContentHtml

Gets or sets a value indicating whether items are bound to plain text or HTML.

Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that determines whether users can edit the text in the nodes.

When the **`isReadOnly`** property is set to false, users may edit the content of the tree nodes by typing directly into the nodes. The F2 key can also be used to enter edit mode with the whole node content selected.

You may customize the editing behavior using the following methods and events:

Methods: **`startEditing`**, **`finishEditing`**.

Events: **`nodeEditStarting`**, **`nodeEditStarted`**, **`nodeEditEnding`**, **`nodeEditEnded`**.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemsSource

Gets or sets the array that contains the **`TreeView`** items.

`TreeView` #see:itemsSource arrays usually have a hierarchical structure with items that contain child items. There is no fixed limit to the depth of the items.

For example, the array below would generate a tree with three top-level nodes, each with two child nodes:

```
var tree = new wijmo.input.TreeView('#treeView', {
    displayMemberPath: 'header',
    childItemsPath: 'items',
    itemsSource: [
        { header: '1 first', items: [
            { header: '1.1 first child' },
            { header: '1.2 second child' },
        ] },
        { header: '2 second', items: [
            { header: '3.1 first child' },
            { header: '3.2 second child' },
        ] },
        { header: '3 third', items: [
            { header: '3.1 first child' },
            { header: '3.2 second child' },
        ] }
    ]
});
```

Type
any[]

lazyLoadFunction

Gets or sets a function that loads child nodes on demand.

The **lazyLoadFunction** takes two parameters: the node being expanded and a callback to be invoked when the data becomes available.

The callback function tells the **TreeView** that the node loading process has been completed. It should always be called, even if there are errors when loading the data.

For example:

```
var treeViewLazyLoad = new wijmo.input.TreeView('#treeViewLazyLoad', {
   displayMemberPath: 'header',
   childItemsPath: 'items',
   itemsSource: [ // start with three lazy-loaded nodes
       { header: 'Lazy Node 1', items: []},
       { header: 'Lazy Node 2', items: [] },
       { header: 'Lazy Node 3', items: [] }
   ],
   lazyLoadFunction: function (node, callback) {
       setTimeout(function () { // simulate http delay
           var result = [ // simulate result
               { header: 'Another lazy node...', items: [] },
               { header: 'A non-lazy node without children' },
               { header: 'A non-lazy node with child nodes', items: [
                 { header: 'hello' },
                 { header: 'world' }
               ]}
           ];
           callback(result); // return result to control
       }, 2500); // simulated 2.5 sec http delay
   }
});
```

Trees with lazy-loaded nodes have some restrictions: their nodes may not have checkboxes (see the **showCheckboxes** property) and the **collapseToLevel** method will not expand collapsed nodes that have not been loaded yet.

Type
Function

nodes

Gets an array of **TreeNode** objects representing the nodes currently loaded.

Type
TreeNode[]

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedItem

Gets or sets the data item that is currently selected.

Type
any

selectedNode

Gets or sets the **`TreeNode`** that is currently selected.

Type
`TreeNode`

selectedPath

Gets an array containing the text of all nodes from the root to the currently selected node.

Type
string[]

showCheckboxes

Gets or sets a value that determines whether the **`TreeView`** should add checkboxes to nodes and manage their state.

This property can be used only on trees without lazy-loaded nodes (see the **`lazyLoadFunction`** property).

See also the **`checkedItems`** property and **`checkedItemsChanged`** event.

Type
boolean

totalItemCount

Gets the total number of items in the tree.

Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

checkAllItems

```
checkAllItems(check: boolean): void
```

Checks or unchecks all checkboxes on the tree.

Parameters

- **check**: **boolean**
  Whether to check or unckeck all checkboxes.

Returns
void

collapseToLevel

```
collapseToLevel(level: number): void
```

Collapses all the tree items to a given level.

This method will typically expand or collapse multiple nodes at once. But it will not perform lazy-loading on any nodes, so collapsed nodes that must be lazy-loaded will not be expanded.

Parameters

- **level**: **number**
  Maximum node level to show.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
`Control`
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Returns
boolean

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getFirstNode

```
getFirstNode(visible?: boolean, enabled?: boolean): TreeNode
```

Gets a reference to the first **`TreeNode`** in the **`TreeView`**.

Parameters

- **visible**: **boolean** OPTIONAL
  Whether to return only visible nodes (whose ancestors are not collapsed).

- **enabled**: **boolean** OPTIONAL
  Whether to return only enabled nodes (whose ancestors are not disabled).

Returns
`TreeNode`

getLastNode

```
getLastNode(visible?: boolean, enabled?: boolean): TreeNode
```

Gets a reference to the last **`TreeNode`** in the **`TreeView`**.

Parameters

- **visible**: **boolean** OPTIONAL
  Whether to return only visible nodes (whose ancestors are not collapsed).

- **enabled**: **boolean** OPTIONAL
  Whether to return only enabled nodes (whose ancestors are not disabled).

Returns
`TreeNode`

getNode

```
getNode(item: any): TreeNode
```

Gets the **TreeNode** object representing a given data item.

Parameters

- **item**: **any**
  The data item to look for.

Returns
TreeNode

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

loadTree

```
loadTree(): void
```

Loads the tree using data from the current **itemsSource**.

Returns
void

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

Raises the **checkedItemsChanged** event.

Parameters
- **e**: **EventArgs** OPTIONAL

Returns
void

onDragEnd

```
onDragEnd(e?: EventArgs): void
```

Raises the **dragEnd** event.

Parameters
- **e**: **EventArgs** OPTIONAL

Returns
void

onDragOver

```
onDragOver(e: TreeNodeDragDropEventArgs): boolean
```

Raises the **dragOver** event.

Parameters

- **e**: **TreeNodeDragDropEventArgs**
  TreeNodeDragDropEventArgs that contains the event data.

Returns
boolean

onDragStart

```
onDragStart(e: TreeNodeEventArgs): boolean
```

Raises the **dragStart** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
boolean

onDrop

```
onDrop(e: TreeNodeDragDropEventArgs): boolean
```

Raises the **drop** event.

Parameters

- **e**: **TreeNodeDragDropEventArgs**
  TreeNodeDragDropEventArgs that contains the event data.

Returns
boolean

onFormatItem

```
onFormatItem(e: FormatNodeEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatNodeEventArgs**
  FormatNodeEventArgs that contains the event data.

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsCheckedChanged

```
onIsCheckedChanged(e: TreeNodeEventArgs): void
```

Raises the **isCheckedChanged** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
void

onIsCheckedChanging

```
onIsCheckedChanging(e: TreeNodeEventArgs): boolean
```

Raises the **isCheckedChanging** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
boolean

onIsCollapsedChanged

```
onIsCollapsedChanged(e: TreeNodeEventArgs): void
```

Raises the **isCollapsedChanged** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
void

onIsCollapsedChanging

```
onIsCollapsedChanging(e: TreeNodeEventArgs): boolean
```

Raises the **isCollapsedChanging** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
boolean

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

Raises the **itemClicked** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

Raises the **loadedItems** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadingItems

```
onLoadingItems(e?: EventArgs): void
```

Raises the **loadingItems** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onNodeEditEnded

```
onNodeEditEnded(e: TreeNodeEventArgs): void
```

Raises the **nodeEditEnded** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
void

onNodeEditEnding

```
onNodeEditEnding(e: TreeNodeEventArgs): boolean
```

Raises the **nodeEditEnding** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
boolean

onNodeEditStarted

```
onNodeEditStarted(e: TreeNodeEventArgs): void
```

Raises the **nodeEditStarted** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
void

onNodeEditStarting

```
onNodeEditStarting(e: TreeNodeEventArgs): boolean
```

Raises the **nodeEditStarting** event.

Parameters

- **e**: **TreeNodeEventArgs**
  TreeNodeEventArgs that contains the event data.

Returns
boolean

onSelectedItemChanged

```
onSelectedItemChanged(e?: EventArgs): void
```

Raises the **selectedItemChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(): void
```

Overridden to re-populate the tree.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

startEditing

```
startEditing(node?: TreeNode): boolean
```

Starts editing a given **`TreeNode`**.

Parameters

- **node**: **TreeNode** OPTIONAL
  `TreeNode` to edit. If not provided, the currently selected node is used.

Returns
boolean

Events

checkedItemsChanged

Occurs when the value of the **`checkedItems`** property changes.

Arguments
`EventArgs`

dragEnd

Occurs when the user finishes a drag/drop operation, either by dropping a node into a new location or by canceling the operation with the mouse or keyboard.

Arguments
`EventArgs`

dragOver

Occurs while the user drags a node over other nodes on the **`TreeView`**.

This event only occurs if the **`allowDragging`** property is set to true.

You may prevent drop operations over certain nodes and/or positions by setting the event's **cancel** parameter to true.

Arguments
`TreeNodeDragDropEventArgs`

dragStart

Occurs when the user starts dragging a node.

This event only occurs if the **`allowDragging`** property is set to true.

You may prevent nodes from being dragged by setting the event's **cancel** parameter to true.

Arguments
`TreeNodeEventArgs`

drop

Occurs when the user drops a on the **TreeView**.

Arguments
TreeNodeDragDropEventArgs

formatItem

Occurs when an element representing a node has been created.

This event can be used to format nodes for display.

The example below uses the **formatItem** event to add a "new" badge to the right of new items on the tree.

```
var treeViewFmtItem = new wijmo.input.TreeView('#treeViewFmtItem', {
    displayMemberPath: 'header',
    childItemsPath: 'items',
    itemsSource: items,
    formatItem: function (s, e) {
        if (e.dataItem.newItem) {
            e.element.innerHTML +=
                '<img style="margin-left:6px" src="resources/new.png"/>';
        }
    }
});
```

Arguments
FormatNodeEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

isCheckedChanged

Occurs after the value of the **`isChecked`** property changes.

Arguments
`TreeNodeEventArgs`

isCheckedChanging

Occurs before the value of the **`isChecked`** property changes.

Arguments
`TreeNodeEventArgs`

isCollapsedChanged

Occurs after the value of the **`isCollapsed`** property changes.

Arguments
`TreeNodeEventArgs`

isCollapsedChanging

Occurs before the value of the **`isCollapsed`** property changes.

Arguments
`TreeNodeEventArgs`

itemClicked

Occurs when the user clicks an item or presses the Enter key and an item is selected.

This event is typically used in navigation trees. Use the **`selectedItem`** property to get the item that was clicked.

Arguments
`EventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Arguments
`EventArgs`

loadedItems

Occurs after the tree items have been generated.

Arguments
`EventArgs`

loadingItems

Occurs before the tree items are generated.

Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

nodeEditEnded

Occurs after a **`TreeNode`** has exited edit mode.

Arguments
`TreeNodeEventArgs`

nodeEditEnding

Occurs before a **`TreeNode`** exits edit mode.

Arguments
`TreeNodeEventArgs`

nodeEditStarted

Occurs after a **`TreeNode`** has entered edit mode.

Arguments
`TreeNodeEventArgs`

nodeEditStarting

Occurs before a **`TreeNode`** enters edit mode.

Arguments
`TreeNodeEventArgs`

selectedItemChanged

Occurs when the value of the **`selectedItem`** property changes.

Arguments
`EventArgs`

DropPosition Enum

File
`wijmo.nav.js`
Module
wijmo.nav

Specifies the position where a **TreeNode** is being dropped during a drag and drop operation.

Members

Name	Value	Description
Before	0	The node will become the previous sibling of the target node.
After	1	The node will become the next sibling of the target node.
Into	2	The node will become the last child of the target node.

wijmo.grid.sheet Module

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Defines the **FlexSheet** control and associated classes.

The **FlexSheet** control extends the **FlexGrid** control to provide Excel-like features.

Example

Show me (http://jsfiddle.net/Wijmo5/t8tp9tnx)

Classes

- ColumnSortDescription
- DefinedName
- DraggingRowColumnEventArgs
- FlexSheet
- FlexSheetPanel
- HeaderRow
- RowColumnChangedEventArgs
- Sheet
- SheetCollection
- SortManager
- UndoStack
- UnknownFunctionEventArgs

Interfaces

- ICellStyle
- IFormatState

ColumnSortDescription Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Describes a **FlexSheet** column sorting criterion.

Constructor

- constructor

Properties

- ascending
- columnIndex

Methods

- clone

Constructor

constructor

```
constructor(columnIndex: number, ascending: boolean): ColumnSortDescription
```

Initializes a new instance of the **ColumnSortDescription** class.

Parameters

- **columnIndex**: **number**
  Indicates which column to sort the rows by.

- **ascending**: **boolean**
  The sort order.

Returns
ColumnSortDescription

Properties

ascending

Gets or sets the ascending.

Type
boolean

columnIndex

Gets or sets the column index.

Type
number

Methods

clone

```
clone(): ColumnSortDescription
```

Creates a copy of the ColumnSortDescription.

Returns
ColumnSortDescription

DefinedName Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Represents a defined name item of FlexSheet.

Constructor

- constructor

Properties

- name
- sheetName
- value

Constructor

constructor

```
constructor(owner: FlexSheet, name: string, value: any, sheetName?: string): DefinedName
```

Initializes a new instance of the DefinedName class.

Parameters

- **owner**: **FlexSheet**
  The owner @see: FlexSheet control.
- **name**: **string**
  The name of the defined name item.
- **value**: **any**
  The value of the defined name item.
- **sheetName**: **string** OPTIONAL
  The sheet name indicates the defined name item works in which sheet of FlexSheet. If omitted, the defined name item works in all sheets of FlexSheet.

Returns
DefinedName

Properties

name

Gets or sets the name of the defined name item.

Type
string

sheetName

Gets the sheetName of the defined name item.

Type
string

value

Gets or sets the value of the defined name item.

Type
any

DraggingRowColumnEventArgs Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
EventArgs

Provides arguments for the **draggingRowColumn** event.

Constructor

- constructor

Properties

- empty
- isDraggingRows
- isShiftKey

Constructor

constructor

```
constructor(isDraggingRows: boolean, isShiftKey: boolean): DraggingRowColumnEventArgs
```

Initializes a new instance of the **DraggingRowColumnEventArgs** class.

Parameters

- **isDraggingRows**: **boolean**
  Indicates whether the dragging event is triggered due to dragging rows or columns.
- **isShiftKey**: **boolean**
  Indicates whether the shift key is pressed when dragging.

Returns
DraggingRowColumnEventArgs

Properties

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

isDraggingRows

Gets a value indicating whether the event refers to dragging rows or columns.

Type
boolean

isShiftKey

Gets a value indicating whether the shift key is pressed.

Type
boolean

FlexSheet Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
FlexGrid
Derived Classes
WjFlexSheet

Defines the **FlexSheet** control.

The **FlexSheet** control extends the **FlexGrid** control to provide Excel-like features such as a calculation engine, multiple sheets, undo/redo, and XLSX import/export.

A complete list of the functions supported by the **FlexSheet**'s calculation engine can be found here: FlexSheet Functions.

Example

Show me (http://jsfiddle.net/Wijmo5/t8tp9tnx)

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- deferResizing
- definedNames
- editableCollectionView
- editRange
- frozenColumns
- frozenRows
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isFunctionListOpen
- isReadOnly
- isTabHolderVisible
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selectedSheet
- selectedSheetIndex
- selection
- selectionMode
- sheets
- showAlternatingRows
- showDropDown
- showErrors
- showFilterIcons
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortManager
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- undoStack
- validateEdits
- viewRange
- virtualizationThreshold

Methods

- addBoundSheet
- addEventListener
- addFunction

- addUnboundSheet
- applyCellsStyle
- applyFunctionToCell
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- clear
- collapseGroupsToLevel
- containsFocus
- convertNumberToAlpha
- deferUpdate
- deleteColumns
- deleteRows
- dispose
- disposeAll
- endUpdate
- evaluate
- finishEditing
- focus
- freezeAtCursor
- getCellBoundingRect
- getCellData
- getCellValue
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getSelectionFormatState
- getTemplate
- hideFunctionList
- hitTest
- initialize
- insertColumns
- insertRows
- invalidate
- invalidateAll
- isRangeValid
- load
- loadAsync
- mergeRange
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onColumnChanged
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowColumn
- onDraggingRowOver
- onDroppingRowColumn
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoaded
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onPrepareChangingColumn
- onPrepareChangingRow
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowChanged
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectedSheetChanged
- onSelectionChanged
- onSelectionChanging
- onSheetCleared
- onSortedColumn
- onSortingColumn
- onUnknownFunction
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- redo
- refresh

- refreshAll
- refreshCells
- removeEventListener
- save
- saveAsync
- scrollIntoView
- select
- selectNextFunction
- selectPreviousFunction
- setCellData
- setClipString
- showColumnFilter
- showFunctionList
- startEditing
- toggleDropDownList
- undo

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- columnChanged
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowColumn
- draggingRowOver
- droppingRowColumn
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- loaded
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- prepareChangingColumn
- prepareChangingRow
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowChanged
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectedSheetChanged
- selectionChanged
- selectionChanging
- sheetCleared
- sortedColumn
- sortingColumn
- unknownFunction
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): FlexSheet
```

Initializes a new instance of the **FlexSheet** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a jQuery selector (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
FlexSheet

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Inherited From
FlexGrid
Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **isReadOnly** property is set to true.

Inherited From
FlexGrid
Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Inherited From
`FlexGrid`
Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Inherited From
`FlexGrid`
Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Inherited From
`FlexGrid`
Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Inherited From
`FlexGrid`
Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Inherited From
`FlexGrid`
Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **itemsSource**.

The column generation depends on the **itemsSource** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **autoGenerateColumns** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Inherited From
FlexGrid
Type
boolean

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Inherited From
FlexGrid
Type
AutoSizeMode

bottomLeftCells

Gets the **GridPanel** that contains the bottom left cells.

The **bottomLeftCells** panel appears below the row headers, to the left of the **columnFooters** panel.

Inherited From
FlexGrid
Type
GridPanel

cellFactory

Gets or sets the **CellFactory** that creates and updates cells for this grid.

Inherited From
FlexGrid
Type
CellFactory

cells

Gets the **GridPanel** that contains the data cells.

Inherited From
FlexGrid
Type
GridPanel

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Inherited From
`FlexGrid`
Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Inherited From
`FlexGrid`
Type
`Size`

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Inherited From
`FlexGrid`
Type
boolean

collectionView

Gets the **ICollectionView** that contains the grid data.

Inherited From
FlexGrid
Type
ICollectionView

columnFooters

Gets the **GridPanel** that contains the column footer cells.

The **columnFooters** panel appears below the grid cells, to the right of the **bottomLeftCells** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **columnFooters** panel to display summary data for columns that have the **aggregate** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Inherited From
FlexGrid
Type
GridPanel

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Inherited From
FlexGrid
Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Inherited From
`FlexGrid`
Type
string

columns

Gets the grid's column collection.

Inherited From
`FlexGrid`
Type
`ColumnCollection`

controlRect

Gets the bounding rectangle of the control in page coordinates.

Inherited From
`FlexGrid`
Type
`Rect`

STATIC controlTemplate

Overrides the template used to instantiate `FlexSheet` control.

Type
any

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **deferResizing** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Inherited From
`FlexGrid`
Type
boolean

definedNames

Gets an array the **IDefinedName** objects representing named ranges/expressions defined in the **FlexSheet**.

Type
`ObservableArray`

editableCollectionView

Gets the **IEditableCollectionView** that contains the grid data.

Inherited From
`FlexGrid`
Type
`IEditableCollectionView`

editRange

Gets a **CellRange** that identifies the cell currently being edited.

Inherited From
`FlexGrid`
Type
`CellRange`

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Inherited From
`FlexGrid`
Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Inherited From
`FlexGrid`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isFunctionListOpen

Gets a value indicating whether the function list is opened.

Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Inherited From
`FlexGrid`
Type
boolean

isTabHolderVisible

Gets or sets a value indicating whether the TabHolder is visible.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Inherited From
FlexGrid
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** that contains items shown on the grid.

Inherited From
FlexGrid
Type
any

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **getError** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Inherited From
FlexGrid
Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **MoveDown**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Inherited From
`FlexGrid`
Type
`KeyAction`

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **None**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **Cycle**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **CycleOut** setting that causes the selection to move through the cells (as **Cycle**), and then on to the next/previous control on the page when the last or first cells are selected.

Inherited From
`FlexGrid`
Type
`KeyAction`

mergeManager

Gets or sets the **MergeManager** object responsible for determining how cells should be merged.

Inherited From
`FlexGrid`
Type
`MergeManager`

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **`newRowAtTop`** property to true, and you want the new row template to remain visible at all times, set the **`frozenRows`** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **`allowAddNew`** property is set to true and if the **`itemsSource`** object supports adding new items.

Inherited From
`FlexGrid`
Type
boolean

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **`preserveOutlineState`** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Inherited From
`FlexGrid`
Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Inherited From
`FlexGrid`
Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Inherited From
`FlexGrid`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Inherited From
`FlexGrid`
Type
string

rowHeaders

Gets the **`GridPanel`** that contains the row header cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

rows

Gets the grid's row collection.

Inherited From
`FlexGrid`
Type
`RowCollection`

scrollPosition

Gets or sets a **`Point`** that represents the value of the grid's scrollbars.

Inherited From
`FlexGrid`
Type
`Point`

scrollSize

Gets the size of the grid content in pixels.

Inherited From
`FlexGrid`
Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when **`selectionMode`** is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when **`selectionMode`** is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedSheet

Gets the current **`Sheet`** in the **FlexSheet**.

Type
`Sheet`

selectedSheetIndex

Gets or sets the index of the current sheet in the **`FlexSheet`**.

Type
number

selection

Gets or sets the current selection.

Inherited From
`FlexGrid`
Type
`CellRange`

selectionMode

Gets or sets the current selection mode.

Inherited From
`FlexGrid`
Type
`SelectionMode`

sheets

Gets the collection of **`Sheet`** objects representing workbook sheets.

Type
`SheetCollection`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Inherited From
`FlexGrid`
Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **`showDropDown`** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`FlexGrid`
Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **`itemValidator`** property or the **`getError`** property on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showFilterIcons

Gets or sets the visiblity of the filter icon.

Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Inherited From
`FlexGrid`
Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

sortManager

Gets the **`SortManager`** instance that controls **FlexSheet** sorting.

Type
`SortManager`

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Inherited From
`FlexGrid`
Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Inherited From
`FlexGrid`
Type
boolean

topLeftCells

Gets the **`GridPanel`** that contains the top left cells (to the left of the column headers).

Inherited From
`FlexGrid`
Type
`GridPanel`

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Inherited From
`FlexGrid`
Type
number

undoStack

Gets the **`UndoStack`** instance that controls undo and redo operations of the **FlexSheet**.

Type
`UndoStack`

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **`getError`** method on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

viewRange

Gets the range of cells currently in view.

Inherited From
`FlexGrid`
Type
`CellRange`

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Inherited From
`FlexGrid`
Type
number

Methods

addBoundSheet

```
addBoundSheet(sheetName: string, source: any, pos?: number, grid?: FlexGrid): Sheet
```

Add a bound **Sheet** to the **FlexSheet**.

Parameters

- **sheetName**: **string**
  The name of the **Sheet**.
- **source**: **any**
  The items source for the **Sheet**.
- **pos**: **number** OPTIONAL
  The position in the **sheets** collection.
- **grid**: **FlexGrid** OPTIONAL
  The **FlexGrid** instance associated with the **Sheet**. If not specified then new **FlexGrid** instance will be created.

Returns
Sheet

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

addFunction

```
addFunction(name: string, func: Function, description?: string, minParamsCount?: number, maxParamsCount?: number): void
```

Add custom function in **FlexSheet**.

Parameters

- **name**: **string**
  the name of the custom function.

- **func**: **Function**
  the custom function.
  The function signature looks as follows:

  ```
  function (...params: any[][][]): any;
  ```

  The function takes a variable number of parameters, each parameter corresponds to an expression passed as a function argument. Independently of whether the expression passed as a function argument resolves to a single value or to a cell range, each parameter value is always a two dimensional array of resolved values. The number of rows (first index) and columns (second index) in the array corresponds to the size of the specified cell range. In case where argument is an expression that resolves to a single value, it will be a one-to-one array where its value can be retrieved using the param[0][0] indexer.
  The sample below adds a custom Sum Product function ('customSumProduct') to the FlexSheet:

  ```
  flexSheet.addFunction('customSumProduct', (...params: any[][][]) => {
     let result = 0,
         range1 = params[0],
         range2 = params[1];

     if (range1.length > 0 && range1.length === range2.length && range1[0].length === range2[0].length) {
         for (let i = 0; i < range1.length; i++) {
             for (let j = 0; j < range1[0].length; j++) {
                 result += range1[i][j] * range2[i][j];
             }
         }
     }
     return result;
  }, 'Custom SumProduct Function', 2, 2);
  ```

  After adding this function, it can be used it in sheet cell expressions, like here:

  ```
  =customSumProduct(A1:B5, B1:C5)
  ```

- **description**: **string** OPTIONAL
  the description of the custom function, it will be shown in the function autocompletion of the **FlexSheet**.

- **minParamsCount**: **number** OPTIONAL

the minimum count of the parameter that the function need.

- **maxParamsCount**: **number** OPTIONAL
  the maximum count of the parameter that the function need. If the count of the parameters in the custom function is arbitrary, the minParamsCount and maxParamsCount should be set to null.

Returns
void

addUnboundSheet

```
addUnboundSheet(sheetName?: string, rows?: number, cols?: number, pos?: number, grid?: FlexGrid): Sheet
```

Add an unbound **Sheet** to the **FlexSheet**.

Parameters

- **sheetName**: **string** OPTIONAL
  The name of the Sheet.
- **rows**: **number** OPTIONAL
  The row count of the Sheet.
- **cols**: **number** OPTIONAL
  The column count of the Sheet.
- **pos**: **number** OPTIONAL
  The position in the **sheets** collection.
- **grid**: **FlexGrid** OPTIONAL
  The **FlexGrid** instance associated with the **Sheet**. If not specified then new **FlexGrid** instance will be created.

Returns
Sheet

applyCellsStyle

```
applyCellsStyle(cellStyle: ICellStyle, cells?: CellRange[], isPreview?: boolean): void
```

Apply the style to a range of cells.

Parameters

- **cellStyle**: **ICellStyle**
  The **ICellStyle** object to apply.
- **cells**: **CellRange**[] OPTIONAL
  An array of **CellRange** objects to apply the style to. If not specified then style is applied to the currently selected cells.
- **isPreview**: **boolean** OPTIONAL
  Indicates whether the applied style is just for preview.

Returns
void

applyFunctionToCell

```
applyFunctionToCell(): void
```

Inserts the selected function from the function list to the cell value editor.

Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.

- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`Control`
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
FlexGrid
Returns
void

clear

```
clear(): void
```

Clears the content of the **FlexSheet** control.

Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Inherited From
FlexGrid
Returns
void

containsFocus

```
containsFocus(): boolean
```

Overrides the base class method to take into account the function list.

Returns
boolean

STATIC convertNumberToAlpha

```
convertNumberToAlpha(c: number): string
```

Converts the number value to its corresponding alpha value. For instance: 0, 1, 2...to a, b, c...

Parameters

- **c**: **number**
  The number value need to be converted.

Returns
string

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

deleteColumns

```
deleteColumns(index?: number, count?: number): void
```

Deletes columns from the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The starting index of the deleting columns. If not specified then columns will be deleted starting from the first column of the current selection.

- **count**: **number** OPTIONAL
  The numbers of columns to delete. If not specified then one column will be deleted.

Returns
void

deleteRows

```
deleteRows(index?: number, count?: number): void
```

Deletes rows from the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The starting index of the deleting rows. If not specified then rows will be deleted starting from the first row of the current selection.

- **count**: **number** OPTIONAL
  The numbers of rows to delete. If not specified then one row will be deleted.

Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

evaluate

```
evaluate(formula: string, format?: string, sheet?: Sheet): any
```

Evaluates a formula.

FlexSheet formulas follow the Excel syntax, including a large subset of the functions supported by Excel. A complete list of the functions supported by **FlexSheet** can be found here: **FlexSheet Functions**.

Parameters

- **formula**: **string**
  The formula to evaluate. The formula may start with an optional equals sign ('=').
- **format**: **string** OPTIONAL
  If specified, defines the .Net format that will be applied to the evaluated value.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose data will be used for evaluation. If not specified then the current sheet is used.

Returns
any

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Inherited From
FlexGrid
Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Inherited From
FlexGrid
Returns
void

freezeAtCursor

```
freezeAtCursor(): void
```

Freeze or unfreeze the columns and rows of the **FlexSheet** control.

Returns
void

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
`FlexGrid`
Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Inherited From
`FlexGrid`
Returns
any

getCellValue

```
getCellValue(rowIndex: number, colIndex: number, formatted?: boolean, sheet?: Sheet): any
```

Gets the evaluated cell value.

Unlike the **getCellData** method that returns a raw data that can be a value or a formula, the **getCellValue** method always returns an evaluated value, that is if the cell contains a formula then it will be evaluated first and the resulting value will be returned.

Parameters

- **rowIndex**: **number**
  The row index of the cell.
- **colIndex**: **number**
  The column index of the cell.
- **formatted**: **boolean** OPTIONAL
  Indicates whether to return an original or a formatted value of the cell.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose value to evaluate. If not specified then the data from current sheet is used.

Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

FlexSheet overrides this method to support multiple rows or columns selection in **FlexSheet**.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Inherited From
`FlexGrid`
Returns
`Column`

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getMergedRange

```
getMergedRange(panel: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**. This method overrides the getMergedRange method of its parent class FlexGrid

Parameters

- **panel**: **GridPanel**
  GridPanel that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.
- **c**: **number**
  Column index of the cell to inspect.

Inherited From
FlexGrid
Returns
SelectedState

getSelectionFormatState

```
getSelectionFormatState(): IFormatState
```

Gets the **`IFormatState`** object describing formatting of the selected cells.

Returns
`IFormatState`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hideFunctionList

```
hideFunctionList(): void
```

Close the function list.

Returns
void

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Inherited From
FlexGrid
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

insertColumns

```
insertColumns(index?: number, count?: number): void
```

Inserts columns in the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The position where new columns should be added. If not specified then columns will be added before the left column of the current selection.

- **count**: **number** OPTIONAL
  The numbers of columns to add. If not specified then one column will be added.

Returns
void

insertRows

```
insertRows(index?: number, count?: number): void
```

Inserts rows in the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The position where new rows should be added. If not specified then rows will be added before the first row of the current selection.

- **count**: **number** OPTIONAL
  The numbers of rows to add. If not specified then one row will be added.

Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Inherited From
`FlexGrid`
Returns
boolean

load

```
load(workbook: any): void
```

Loads the workbook into **FlexSheet**. This method works with JSZip 2.5.

For example:

```
// This sample opens an xlsx file chosen through Open File
// dialog and fills FlexSheet


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>
<div id="flexHost"></>


// JavaScript
var flexSheet = new wijmo.grid.FlexSheet("#flexHost"),
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
            flexSheet.load(reader.result);
        };
        reader.readAsArrayBuffer(file);
    }
}
```

Parameters

- **workbook**: **any**

  A workbook instance or a Blob instance or a base-64 string or an ArrayBuffer containing xlsx file content.

Returns
void

loadAsync

```
loadAsync(workbook: any, onLoaded?: (workbook: wijmo.xlsx.Workbook), onError?: (reason?: any)): void
```

Loads the workbook into **FlexSheet** asynchronously. This method works with JSZip 3.0.

Parameters

- **workbook**: **any**
  A workbook instance or a Blob instance or a base-64 string or an ArrayBuffer containing xlsx file content.

- **onLoaded**: **(workbook: wijmo.xlsx.Workbook)** OPTIONAL
  This callback provides an approach to get the loaded workbook instance. Since this method is an asynchronous method, user is not able to get the loaded workbook instance immediately. User has to get the loaded workbook instance through this callback. This has a single parameter, the loaded workbook instance. It is passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when loading. This has a single parameter, the failure reason. The return value is passed to user if he wants to catch the load failure reason.

  For example:

```
flexsheet.loadAsync(blob, function (workbook) {

     // user can access the loaded workbook instance in this callback.
     var app = worksheet.application ;
     ...
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of load failure is ' + reason);
});
```

Returns
void

mergeRange

```
mergeRange(cells?: CellRange, isCopyMergeCell?: boolean): void
```

Merges the selected **CellRange** into one cell.

Parameters

- **cells**: **CellRange** OPTIONAL
  The **CellRange** to merge.

- **isCopyMergeCell**: **boolean** OPTIONAL
  This parameter indicates that merge operation is done by copy\paste merge cell or not.

Returns
void

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onColumnChanged

```
onColumnChanged(e: RowColumnChangedEventArgs): void
```

Raises the columnChanged event.

Parameters

- **e**: **RowColumnChangedEventArgs**

Returns
void

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRowColumn

```
onDraggingRowColumn(e: DraggingRowColumnEventArgs): void
```

Raises the draggingRowColumn event.

Parameters

- **e**: **DraggingRowColumnEventArgs**

Returns
void

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDroppingRowColumn

```
onDroppingRowColumn(e?: EventArgs): void
```

Raises the droppingRowColumn event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoaded

```
onLoaded(e?: EventArgs): void
```

Raises the loaded event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPrepareChangingColumn

```
onPrepareChangingColumn(): void
```

Raises the prepareChangingColumn event.

Returns
void

onPrepareChangingRow

```
onPrepareChangingRow(): void
```

Raises the prepareChangingRow event.

Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowChanged

```
onRowChanged(e: RowColumnChangedEventArgs): void
```

Raises the rowChanged event.

Parameters

- **e**: **RowColumnChangedEventArgs**

Returns
void

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onSelectedSheetChanged

```
onSelectedSheetChanged(e: PropertyChangedEventArgs): void
```

Raises the currentSheetChanged event.

Parameters

- **e**: **PropertyChangedEventArgs**
  PropertyChangedEventArgs that contains the event data.

Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onSheetCleared

```
onSheetCleared(e?: EventArgs): void
```

Raises the sheetCleared event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUnknownFunction

```
onUnknownFunction(e: UnknownFunctionEventArgs): void
```

Raises the unknownFunction event.

Parameters

- **e**: **UnknownFunctionEventArgs**

Returns
void

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

redo

```
redo(): void
```

Redo the last user action.

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Overridden to refresh the sheet and the TabHolder.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.

- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.

- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Inherited From
FlexGrid
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

save

```
save(fileName?: string): Workbook
```

Saves **FlexSheet** to xlsx file. This method works with JSZip 2.5.

For example:

```
// This sample exports FlexSheet content to an xlsx file.
// click.


// HTML
<button
    onclick="saveXlsx('FlexSheet.xlsx')">
    Save
</button>


// JavaScript
function saveXlsx(fileName) {

    // Save the flexGrid to xlsx file.
    flexsheet.save(fileName);
}
```

Parameters

- **fileName**: **string** OPTIONAL
  Name of the file that is generated.

Returns
Workbook

saveAsync

```
saveAsync(fileName?: string, onSaved?: (base64?: string), onError?: (reason?: any)): void
```

Saves the **FlexSheet** to xlsx file asynchronously. This method works with JSZip 3.0.

Parameters

- **fileName**: **string** OPTIONAL
  Name of the file that is generated.

- **onSaved**: **(base64?: string)** OPTIONAL
  This callback provides an approach to get the base-64 string that represents the content of the saved FlexSheet. Since this method is an asynchronous method, user is not able to get the base-64 string immediately. User has to get the base-64 string through this callback. This has a single parameter, the base64 string of the saved flexsheet. It is passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when saving. This has a single parameter, the failure reason. The return value is passed to user if he wants to catch the save failure reason.

  For example:

```
flexsheet.saveAsync('', function (base64) {

     // user can access the base64 string in this callback.
     document.getElementByID('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64,' + base64;
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of save failure is ' + reason);
});
```

Returns
void

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.
- **c**: **number**
  Index of the column to scroll into view.

Inherited From
FlexGrid
Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

FlexSheet overrides this method to adjust the selection cell range for the merged cells in the **FlexSheet**.

Parameters

- **rng**: **any**
  The cell range to select.
- **show**: **any** OPTIONAL
  Indicates whether to scroll the new selection into view.

Returns
void

selectNextFunction

```
selectNextFunction(): void
```

Select next function in the function list.

Returns
void

selectPreviousFunction

```
selectPreviousFunction(): void
```

Select previous function in the function list.

Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean): boolean
```

Overrides the setCellData function of the base class.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.

Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Override the **setClipString** method of **FlexGrid**.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.
- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Returns
void

showColumnFilter

```
showColumnFilter(): void
```

Show the filter editor.

Returns
void

showFunctionList

```
showFunctionList(target: HTMLElement): void
```

Open the function list.

Parameters

- **target**: **HTMLElement**
  The DOM element that toggle the function list.

Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Inherited From
FlexGrid
Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Inherited From
FlexGrid
Returns
void

undo

```
undo(): void
```

Undo the last user action.

Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Inherited From
`FlexGrid`
Arguments
`CellEditEndingEventArgs`

columnChanged

Occurs after the **FlexSheet** insert\delete columns.

Arguments
RowColumnChangedEventArgs

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **allowDelete** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **`allowDelete`** property).

The event handler may cancel the row deletion.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggedColumn

Occurs when the user finishes dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggedRow

Occurs when the user finishes dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRowColumn

Occurs when dragging the rows or the columns of the **FlexSheet**.

Arguments
`DraggingRowColumnEventArgs`

draggingRowOver

Occurs as the user drags a row to a new position.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

droppingRowColumn

Occurs when dropping the rows or the columns of the **FlexSheet**.

Arguments
`EventArgs`

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **`itemFormatter`** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Inherited From
`FlexGrid`
Arguments
`FormatItemEventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loaded

Occurs after the **`FlexSheet`** loads the **`Workbook`** instance

Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

prepareChangingColumn

Occurs before the `FlexSheet` insert\delete columns.

Arguments
`EventArgs`

prepareChangingRow

Occurs before the `FlexSheet` insert\delete rows.

Arguments
`EventArgs`

resizedColumn

Occurs when the user finishes resizing a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizedRow

Occurs when the user finishes resizing rows.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowChanged

Occurs after the **`FlexSheet`** insert\delete rows.

Arguments
`RowColumnChangedEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **rowEditStarted** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

rowEditStarted

Occurs after a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

selectedSheetChanged

Occurs when current sheet index changed.

Arguments
`PropertyChangedEventArgs`

selectionChanged

Occurs after selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sheetCleared

Occurs when the `FlexSheet` is cleared.

Arguments
`EventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

unknownFunction

Occurs when the **`FlexSheet`** meets the unknown formula.

Arguments
`UnknownFunctionEventArgs`

updatedLayout

Occurs after the grid has updated its internal layout.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatingLayout

Occurs before the grid updates its internal layout.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

FlexSheetPanel Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
GridPanel

Defines the extension of the **GridPanel** class, which is used by **FlexSheet** where the base **FlexGrid** class uses **GridPanel**. For example, the **cells** property returns an instance of this class.

Constructor

- constructor

Properties

- cellType
- columns
- grid
- height
- hostElement
- rows
- viewRange
- width

Methods

- getCellBoundingRect
- getCellData
- getCellElement
- getSelectedState
- setCellData

Constructor

constructor

```
constructor(grid: FlexGrid, cellType: CellType, rows: RowCollection, cols: ColumnCollection, element: HTMLElement): FlexSheetPanel
```

Initializes a new instance of the **FlexSheetPanel** class.

Parameters

- **grid**: **FlexGrid**
  The **FlexGrid** object that owns the panel.
- **cellType**: **CellType**
  The type of cell in the panel.
- **rows**: **RowCollection**
  The rows displayed in the panel.
- **cols**: **ColumnCollection**
  The columns displayed in the panel.
- **element**: **HTMLElement**
  The HTMLElement that hosts the cells in the control.

Returns
FlexSheetPanel

Properties

cellType

Gets the type of cell contained in the panel.

Inherited From
GridPanel
Type
CellType

columns

Gets the panel's column collection.

Inherited From
`GridPanel`
Type
`ColumnCollection`

grid

Gets the grid that owns the panel.

Inherited From
`GridPanel`
Type
`FlexGrid`

height

Gets the total height of the content in this panel.

Inherited From
`GridPanel`
Type
number

hostElement

Gets the host element for the panel.

Inherited From
`GridPanel`
Type
HTMLElement

rows

Gets the panel's row collection.

Inherited From
`GridPanel`
Type
`RowCollection`

viewRange

Gets a **`CellRange`** that indicates the range of cells currently visible on the panel.

Inherited From
`GridPanel`
Type
`CellRange`

width

Gets the total width of the content in the panel.

Inherited From
`GridPanel`
Type
number

Methods

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a cell's bounds in viewport coordinates.

The returned value is a **Rect** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same as those used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  The index of the row that contains the cell.
- **c**: **number**
  The index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
GridPanel
Returns
Rect

getCellData

```
getCellData(r: number, c: any, formatted: boolean): any
```

Gets the value stored in a cell in the panel.

Parameters

- **r**: **number**
  The row index of the cell.
- **c**: **any**
  The index, name, or binding of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Returns
any

getCellElement

```
getCellElement(r: number, c: number): HTMLElement
```

Gets the element that represents a cell within this **GridPanel**.

If the cell is not currently in view, this method returns null.

Parameters

- **r**: **number**
  The index of the row that contains the cell.

- **c**: **number**
  The index of the column that contains the cell.

Inherited From
GridPanel
Returns
HTMLElement

getSelectedState

```
getSelectedState(r: number, c: number, rng: CellRange): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Overrides this method to support multiple selection showSelectedHeaders for **FlexSheet**

Parameters

- **r**: **number**
  Specifies Row index of the cell.

- **c**: **number**
  Specifies Column index of the cell.

- **rng**: **CellRange**
  CellRange that contains the cell that would be included.

Returns
SelectedState

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean): boolean
```

Sets the content of a cell in the panel.

Parameters

- **r**: **number**
  The index of the row that contains the cell.
- **c**: **any**
  The index, name, or binding of the column that contains the cell.
- **value**: **any**
  The value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  A value indicating whether to change the value automatically to match the column's data type.

Returns
boolean

HeaderRow Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
Row

Represents a row used to display column header information for a bound sheet.

Constructor

- constructor

Properties

- allowDragging
- allowMerging
- allowResizing
- collectionView
- cssClass
- dataItem
- grid
- height
- index
- isContentHtml
- isReadOnly
- isSelected
- isVisible
- pos
- renderHeight
- renderSize
- size
- visible
- visibleIndex
- wordWrap

Methods

- onPropertyChanged

Constructor

constructor

```
constructor(): HeaderRow
```

Initializes a new instance of the HeaderRow class.

Returns
HeaderRow

Properties

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Inherited From
`RowCol`
Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Inherited From
`RowCol`
Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Inherited From
`RowCol`
Type
boolean

collectionView

Gets the `ICollectionView` bound to this row or column.

Inherited From
`RowCol`
Type
`ICollectionView`

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Inherited From
`RowCol`
Type
string

dataItem

Gets or sets the item in the data collection that the item is bound to.

Inherited From
`Row`
Type
any

grid

Gets the **`FlexGrid`** that owns the row or column.

Inherited From
`RowCol`
Type
`FlexGrid`

height

Gets or sets the height of the row. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`Row`
Type
number

index

Gets the index of the row or column in the parent collection.

Inherited From
`RowCol`
Type
number

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Inherited From
`RowCol`
Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Inherited From
`RowCol`
Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Inherited From
`RowCol`
Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Inherited From
`RowCol`
Type
boolean

pos

Gets the position of the row or column.

Inherited From
`RowCol`
Type
number

renderHeight

Gets the render height of the row.

The value returned takes into account the row's visibility, default size, and min and max sizes.

Inherited From
`Row`
Type
number

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Inherited From
`RowCol`
Type
number

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`RowCol`
Type
number

visible

Gets or sets a value that indicates whether the row or column is visible.

Inherited From
`RowCol`
Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Inherited From
`RowCol`
Type
number

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Inherited From
`RowCol`
Type
boolean

Methods

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Inherited From
`RowCol`
Returns
void

RowColumnChangedEventArgs Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
EventArgs

Provides arguments for rows or columns changed events.

Constructor

- constructor

Properties

- added
- count
- empty
- index

Constructor

constructor

```
constructor(index: number, count: number, added: boolean): RowColumnChangedEventArgs
```

Initializes a new instance of the **`UnknownFunctionEventArgs`** class.

Parameters

- **index**: **number**
  The start index of the changed rows or columns.
- **count**: **number**
  The added or removed count of the rows or columns.
- **added**: **boolean**
  The value indicates the event is for adding ot removing rows or columns.

Returns
RowColumnChangedEventArgs

Properties

added

Gets the value indicates the event is for adding ot removing rows or columns.

Type
boolean

count

Gets the added or removed count of the rows or columns.

Type
number

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

index

Gets the start index of the changed rows or columns.

Type
number

Sheet Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Derived Classes
WjSheet

Represents a sheet within the **FlexSheet** control.

Constructor

- constructor

Properties

- columnCount
- grid
- itemsSource
- name
- rowCount
- selectionRanges
- tableNames
- visible

Methods

- findTable
- getCellStyle
- onNameChanged
- onVisibleChanged

Events

- nameChanged
- visibleChanged

Constructor

constructor

```
constructor(owner?: FlexSheet, grid?: FlexGrid, sheetName?: string, rows?: number, cols?: number): Sheet
```

Initializes a new instance of the **Sheet** class.

Parameters

- **owner**: **FlexSheet** OPTIONAL
  The owner @see: FlexSheet control.
- **grid**: **FlexGrid** OPTIONAL
  The associated **FlexGrid** control used to store the sheet data. If not specified then the new **FlexGrid** control will be created.
- **sheetName**: **string** OPTIONAL
  The name of the sheet within the **FlexSheet** control.
- **rows**: **number** OPTIONAL
  The row count for the sheet.
- **cols**: **number** OPTIONAL
  The column count for the sheet.

Returns
Sheet

Properties

columnCount

Gets or sets the number of columns in the sheet.

Type
number

grid

Gets the associated **FlexGrid** control used to store the sheet data.

Type
FlexGrid

itemsSource

Gets or sets the array or **`ICollectionView`** for the **`FlexGrid`** instance of the sheet.

Type
any

name

Gets or sets the name of the sheet.

Type
string

rowCount

Gets or sets the number of rows in the sheet.

Type
number

selectionRanges

Gets the selection array.

Type
`ObservableArray`

tableNames

Gets the names of tables render in this sheet.

Type
string[]

visible

Gets or sets the sheet visibility.

Type
boolean

Methods

findTable

```
findTable(rowIndex: number, columnIndex: number): Table
```

Finds the table via the cell location.

Parameters

- **rowIndex**: **number**
  the row index of the specified cell.
- **columnIndex**: **number**
  the column index of the specified cell.

Returns
Table

getCellStyle

```
getCellStyle(rowIndex: number, columnIndex: number): ICellStyle
```

Gets the style of specified cell.

Parameters

- **rowIndex**: **number**
  the row index of the specified cell.
- **columnIndex**: **number**
  the column index of the specified cell.

Returns
ICellStyle

onNameChanged

```
onNameChanged(e: PropertyChangedEventArgs): void
```

Raises the **nameChanged** event.

Parameters
- **e**: **PropertyChangedEventArgs**

Returns
void

onVisibleChanged

```
onVisibleChanged(e: EventArgs): void
```

Raises the **visibleChanged** event.

Parameters
- **e**: **EventArgs**

Returns
void

Events

nameChanged

Occurs after the sheet name has changed.

Arguments
PropertyChangedEventArgs

visibleChanged

Occurs after the visible of sheet has changed.

Arguments
EventArgs

SheetCollection Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
ObservableArray

Defines the collection of the **Sheet** objects.

Constructor

- constructor

Properties

- isUpdating
- selectedIndex

Methods

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getValidSheetName
- hide
- implementsInterface
- indexOf
- insert
- isValidSheetName
- onCollectionChanged
- onSelectedSheetChanged
- onSheetCleared
- onSheetNameChanged
- onSheetVisibleChanged
- push
- remove
- removeAt
- selectFirst
- selectLast
- selectNext
- selectPrevious
- setAt
- show
- slice
- sort
- splice

Events

- collectionChanged
- selectedSheetChanged
- sheetCleared
- sheetNameChanged
- sheetVisibleChanged

Constructor

constructor

```
constructor(data?: any[]): ObservableArray
```

Initializes a new instance of the **ObservableArray** class.

Parameters

- **data**: **any**[] OPTIONAL
  Array containing items used to populate the **ObservableArray**.

Inherited From
ObservableArray
Returns
ObservableArray

Properties

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **beginUpdate** and **endUpdate**).

Inherited From
ObservableArray
Type

selectedIndex

Gets or sets the index of the currently selected sheet.

Type
number

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
ObservableArray
Returns
void

clear

```
clear(): void
```

Clear the SheetCollection.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **`beginUpdate`**.

Inherited From
`ObservableArray`
Returns
void

getValidSheetName

```
getValidSheetName(currentSheet: Sheet): string
```

Gets the valid name for the sheet.

Parameters

- **currentSheet**: **Sheet**
  The **`Sheet`** need get the valid name.

Returns
string

hide

```
hide(pos: number): boolean
```

Hides the sheet at the specified position.

Parameters

- **pos**: **number**
  The position of the sheet to hide.

Returns
boolean

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
ObservableArray
Returns
boolean

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

Searches for an item in the array.

Parameters

- **searchElement**: **any**
  Element to locate in the array.

- **fromIndex**: **number** OPTIONAL
  The index where the search should start.

Inherited From
ObservableArray
Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array. Overrides the insert method of its base class **`ObservableArray`**.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Returns
void

isValidSheetName

```
isValidSheetName(sheet: Sheet): boolean
```

Checks whether the sheet name is valid.

Parameters

- **sheet**: **Sheet**
  The **`Sheet`** for which the name needs to check.

Returns
boolean

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
ObservableArray
Returns
void

onSelectedSheetChanged

```
onSelectedSheetChanged(e: PropertyChangedEventArgs): void
```

Raises the **currentChanged** event.

Parameters

- **e**: **PropertyChangedEventArgs**
  PropertyChangedEventArgs that contains the event data.

Returns
void

onSheetCleared

```
onSheetCleared(): void
```

Raises the sheetCleared event.

Returns
void

onSheetNameChanged

```
onSheetNameChanged(e: NotifyCollectionChangedEventArgs): void
```

Raises the **sheetNameChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs**

Returns
void

onSheetVisibleChanged

```
onSheetVisibleChanged(e: NotifyCollectionChangedEventArgs): void
```

Raises the **sheetVisibleChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs**

Returns
void

push

```
push(...item: any[]): number
```

Adds one or more items to the end of the array. Overrides the push method of its base class **`ObservableArray`**.

Parameters

- **...item**: **any**[]
  One or more items to add to the array.

Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
`ObservableArray`
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array. Overrides the removeAt method of its base class **`ObservableArray`**.

Parameters

- **index**: **number**
  Position of the item to remove.

Returns
void

selectFirst

```
selectFirst(): boolean
```

Selects the first sheet in the **`FlexSheet`** control.

Returns
boolean

selectLast

```
selectLast(): boolean
```

Selects the last sheet in the owner **FlexSheet** control.

Returns
boolean

selectNext

```
selectNext(): boolean
```

Select the next sheet in the owner **FlexSheet** control.

Returns
boolean

selectPrevious

```
selectPrevious(): boolean
```

Selects the previous sheet in the owner **FlexSheet** control.

Returns
boolean

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
`ObservableArray`
Returns
void

show

```
show(pos: number): boolean
```

Unhide and selects the **`Sheet`** at the specified position.

Parameters

- **pos**: **number**
  The position of the sheet to show.

Returns
boolean

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
ObservableArray
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes and/or adds items to the array. Overrides the splice method of its base class **`ObservableArray`**.

Parameters

- **index**: **number**
  Position where items will be added or removed.
- **count**: **number**
  Number of items to remove from the array.
- **item**: **any** OPTIONAL
  Item to add to the array.

Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

selectedSheetChanged

Occurs when the **selectedIndex** property changes.

Arguments
`PropertyChangedEventArgs`

sheetCleared

Occurs when the **`SheetCollection`** is cleared.

Arguments
`EventArgs`

sheetNameChanged

Occurs after the name of the sheet in the collection has changed.

Arguments
`NotifyCollectionChangedEventArgs`

sheetVisibleChanged

Occurs after the visible of the sheet in the collection has changed.

Arguments
`NotifyCollectionChangedEventArgs`

SortManager Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Maintains sorting of the selected **Sheet** of the **FlexSheet**.

Constructor

- constructor

Properties

- sortDescriptions

Methods

- addSortLevel
- cancelSort
- checkSortItemExists
- commitSort
- copySortLevel
- deleteSortLevel
- editSortLevel
- moveSortLevel

Constructor

constructor

```
constructor(owner: FlexSheet): SortManager
```

Initializes a new instance of the **SortManager** class.

Parameters
- **owner**: **FlexSheet**
  The **FlexSheet** control that owns this **SortManager**.

Returns
SortManager

Properties

sortDescriptions

Gets or sets the collection of the sort descriptions represented by the **`ColumnSortDescription`** objects.

Type
`CollectionView`

Methods

addSortLevel

```
addSortLevel(columnIndex?: number, ascending?: boolean): void
```

Adds a blank sorting level to the sort descriptions.

Parameters

- **columnIndex**: **number** OPTIONAL
  The index of the column in the FlexSheet control.
- **ascending**: **boolean** OPTIONAL
  The sort order for the sort level.

Returns
void

cancelSort

```
cancelSort(): void
```

Cancel the current sort descriptions to the FlexSheet control.

Returns
void

checkSortItemExists

```
checkSortItemExists(columnIndex): number
```

Check whether the sort item of specific column exists or not

Parameters

- **columnIndex**:
  The index of the column in the FlexSheet control.

Returns
number

commitSort

```
commitSort(undoable?: boolean): void
```

Commits the current sort descriptions to the FlexSheet control.

Parameters

- **undoable**: **boolean** OPTIONAL
  The boolean value indicating whether the commit sort action is undoable.

Returns
void

copySortLevel

```
copySortLevel(): void
```

Adds a copy of the current sorting level to the sort descriptions.

Returns
void

deleteSortLevel

```
deleteSortLevel(columnIndex?: number): void
```

Removes the current sorting level from the sort descriptions.

Parameters

- **columnIndex**: **number** OPTIONAL
  The index of the column in the FlexSheet control.

Returns
void

editSortLevel

```
editSortLevel(columnIndex?: number, ascending?: boolean): void
```

Updates the current sort level.

Parameters

- **columnIndex**: **number** OPTIONAL
  The column index for the sort level.

- **ascending**: **boolean** OPTIONAL
  The sort order for the sort level.

Returns
void

moveSortLevel

```
moveSortLevel(offset: number): void
```

Moves the current sorting level to a new position.

Parameters

- **offset**: **number**
  The offset to move the current level by.

Returns
void

UndoStack Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Controls undo and redo operations in the **FlexSheet**.

Constructor

- constructor

Properties

- canRedo
- canUndo

Methods

- clear
- onUndoStackChanged
- redo
- undo

Events

- undoStackChanged

Constructor

constructor

```
constructor(owner: FlexSheet): UndoStack
```

Initializes a new instance of the **UndoStack** class.

Parameters

- **owner**: **FlexSheet**
  The **FlexSheet** control that the **UndoStack** works for.

Returns
UndoStack

Properties

canRedo

Checks whether a redo action can be performed.

Type
boolean

canUndo

Checks whether an undo action can be performed.

Type
boolean

Methods

clear

```
clear(): void
```

Clears the undo stack.

Returns
void

onUndoStackChanged

```
onUndoStackChanged(): void
```

Raises the **undoStackChanged** event.

Returns
void

redo

```
redo(): void
```

Redo the last undone action.

Returns
void

undo

```
undo(): void
```

Undo the last action.

Returns
void

Events

undoStackChanged

Occurs after the undo stack has changed.

Arguments
`EventArgs`

UnknownFunctionEventArgs Class

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet
Base Class
EventArgs

Provides arguments for unknown function events.

Constructor

- constructor

Properties

- empty
- funcName
- params
- value

Constructor

constructor

```
constructor(funcName: string, params: any[]): UnknownFunctionEventArgs
```

Initializes a new instance of the **UnknownFunctionEventArgs** class.

Parameters

- **funcName**: **string**
  The name of the unknown function.

- **params**: **any**[]
  The parameters' value list of the nuknown function.

Returns
UnknownFunctionEventArgs

Properties

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

funcName

Gets the name of the unknown function.

Type
string

params

Gets the parameters' value list of the nuknown function.

Type
any[]

value

Gets or sets the result for the unknown funtion.

Type
string

ICellStyle Interface

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Defines the cell styling properties.

Properties

- backgroundColor
- borderBottomColor
- borderBottomStyle
- borderBottomWidth
- borderLeftColor
- borderLeftStyle
- borderLeftWidth
- borderRightColor
- borderRightStyle
- borderRightWidth
- borderTopColor
- borderTopStyle
- borderTopWidth
- className
- color
- fontFamily
- fontSize
- fontStyle
- fontWeight
- format
- textAlign
- textDecoration
- verticalAlign
- whiteSpace

Properties

backgroundColor

The background color.

Type
any

borderBottomColor

Color of the Bottom border.

Type
any

borderBottomStyle

Style of the Bottom border.

Type
string

borderBottomWidth

Width of the Bottom border.

Type
string

borderLeftColor

Color of the Left border.

Type
any

borderLeftStyle

Style of the Left border.

Type
string

borderLeftWidth

Width of the Left border.

Type
string

borderRightColor

Color of the Right border.

Type
any

borderRightStyle

Style of the Right border.

Type
string

borderRightWidth

Width of the Right border.

Type
string

borderTopColor

Color of the Top border.

Type
any

borderTopStyle

Style of the Top border.

Type
string

borderTopWidth

Width of the Top border.

Type
string

className

The CSS class name to add to a cell.

Type
string

color

The font color.

Type
any

fontFamily

The font family.

Type
string

fontSize

The font size.

Type
string

fontStyle

The font style.

Type
string

fontWeight

The font weight.

Type
string

format

Format string for formatting the value of the cell.

Type
string

textAlign

The text alignment.

Type
string

textDecoration

The text decoration.

Type
string

verticalAlign

The vertical alignment.

Type
string

whiteSpace

Describes how whitespace inside the element is handled.

Type
string

IFormatState Interface

File
`wijmo.grid.sheet.js`
Module
wijmo.grid.sheet

Defines the format states for the cells.

Properties

- isBold
- isItalic
- isMergedCell
- isUnderline
- textAlign

Properties

isBold

Indicates whether the bold style is applied.

Type
boolean

isItalic

Indicates whether the italic style is applied.

Type
boolean

isMergedCell

Indicate whether the current selection is a merged cell.

Type
boolean

isUnderline

Indicates whether the underlined style is applied.

Type
boolean

textAlign

Gets the applied text alignment.

Type
string

wijmo.chart.finance Module

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance

Defines the **FinancialChart** control and its associated classes.

Classes

- FinancialChart
- FinancialSeries

Enums

- DataFields
- FinancialChartType
- PointAndFigureScaling
- RangeMode

FinancialChart Class

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance
Base Class
FlexChartCore
Derived Classes
WjFinancialChart

Financial charting control.

Constructor

- constructor

Properties

- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- header
- headerStyle
- hostElement
- interpolateNulls
- isDisabled
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- options
- palette
- plotAreas
- plotMargin
- rightToLeft
- selection
- selectionMode
- series
- symbolSize
- tooltip

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

Constructor

constructor

```
constructor(element: any, options?): FinancialChart
```

Initializes a new instance of the **FlexChart** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Returns
FinancialChart

Properties

axes

Gets the collection of **Axis** objects.

Inherited From
FlexChartCore
Type
ObservableArray

axisX

Gets or sets the main X axis.

Inherited From
FlexChartCore
Type
Axis

axisY

Gets or sets the main Y axis.

Inherited From
`FlexChartCore`
Type
`Axis`

binding

Gets or sets the name of the property that contains the Y values.

Inherited From
`FlexChartCore`
Type
string

bindingX

Gets or sets the name of the property that contains the X data values.

Inherited From
`FlexChartCore`
Type
string

chartType

Gets or sets the type of financial chart to create.

Type
`FinancialChartType`

collectionView

Gets the **ICollectionView** object that contains the chart data.

Inherited From
FlexChartBase
Type
ICollectionView

dataLabel

Gets or sets the point data label.

Inherited From
FlexChartCore
Type
DataLabel

footer

Gets or sets the text displayed in the chart footer.

Inherited From
FlexChartBase
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
FlexChartBase
Type
any

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

interpolateNulls

Gets or sets a value that determines whether to interpolate null values in the data.

If true, the chart interpolates the value of any missing data based on neighboring points. If false, it leaves a break in lines and areas at the points with null values.

Inherited From
`FlexChartCore`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

legendToggle

Gets or sets a value indicating whether clicking legend items toggles the series visibility in the chart.

Inherited From
`FlexChartCore`
Type
boolean

options

Gets or sets various chart options.

The following options are supported:

kagi.fields: Specifies the **`DataFields`** used for the Kagi chart. The default value is DataFields.Close.

kagi.rangeMode: Specifies the **`RangeMode`** for the Kagi chart. The default value is RangeMode.Fixed.

kagi.reversalAmount: Specifies the reversal amount for the Kagi chart. The default value is 14.

```
chart.options = {
  kagi: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     reversalAmount: 14
  }
}
```

lineBreak.newLineBreaks: Gets or sets the number of previous boxes that must be compared before a new box is drawn in Line Break charts. The default value is 3.

```
chart.options = {
  lineBreak: { newLineBreaks: 3 }
}
```

renko.fields: Specifies the **`DataFields`** used for the Renko chart. The default value is DataFields.Close.

renko.rangeMode: Specifies the **`RangeMode`** for the Renko chart. The default value is RangeMode.Fixed.

renko.boxSize: Specifies the box size for the Renko chart. The default value is 14.

```
chart.options = {
  renko: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     boxSize: 14
  }
}
```

Type
any

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **Palettes** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
FlexChartBase
Type
string[]

plotAreas

Gets the collection of **PlotArea** objects.

Inherited From
FlexChartCore
Type
PlotAreaCollection

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selection

Gets or sets the selected chart series.

Inherited From
`FlexChartCore`
Type
`SeriesBase`

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

series

Gets the collection of **`Series`** objects.

Inherited From
`FlexChartCore`
Type
`ObservableArray`

symbolSize

Gets or sets the size of the symbols used for all Series objects in this **`FlexChart`**.

This property may be overridden by the symbolSize property on each **`Series`** object.

Inherited From
`FlexChartCore`
Type
number

tooltip

Gets the chart **Tooltip** object.

The tooltip content is generated using a template that may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

To modify the template, assign a new value to the tooltip's content property. For example:

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

You can disable chart tooltips by setting the template to an empty string.

You can also use the **tooltip** property to customize tooltip parameters such as **showDelay** and **hideDelay**:

```
chart.tooltip.showDelay = 1000;
```

See **ChartTooltip** properties for more details and options.

Inherited From
FlexChartCore
Type
ChartTooltip

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Converts a **Point** from data coordinates to control coordinates.

Parameters

- **pt**: **any**
  Point in data coordinates, or X coordinate of a point in data coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Inherited From
FlexChartCore
Returns
Point

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

Raises the **seriesVisibilityChanged** event.

Parameters

- **e**: **SeriesEventArgs**
  The **SeriesEventArgs** object that contains the event data.

Inherited From
FlexChartCore
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
`FlexChartBase`
Returns
`Point`

pointToData

```
pointToData(pt: any, y?: number): Point
```

Converts a **`Point`** from control coordinates to chart data coordinates.

Parameters

- **pt**: **any**
  The point to convert, in control coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

seriesVisibilityChanged

Occurs when the series visibility changes, for example when the legendToggle property is set to true and the user clicks the legend.

Inherited From
`FlexChartCore`
Arguments
`SeriesEventArgs`

FinancialSeries Class

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance
Base Class
SeriesBase
Derived Classes
WjFinancialChartSeries

Represents a series of data points to display in the chart.

The **Series** class supports all basic chart types. You may define a different chart type on each **Series** object that you add to the **FlexChart** series collection. This overrides the **chartType** property set on the chart that is the default for all **Series** objects in its collection.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SeriesBase
```

Initializes a new instance of the **SeriesBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
SeriesBase
Returns
SeriesBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart. Please note that ColumnVolume, EquiVolume, CandleVolume and ArmsCandleVolume chart types are not supported and should be set on the **`FinancialChart`**.

Type
`FinancialChartType`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

DataFields Enum

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance

Specifies which fields are to be used for calculation. Applies to Renko and Kagi chart types.

Members

Name	Value	Description
Close	0	Close values are used for calculations.
High	1	High values are used for calculations.
Low	2	Low values are used for calculations.
Open	3	Open values are used for calculations.
HighLow	4	High-Low method is used for calculations. DataFields.HighLow is currently not supported with Renko chart types.
HL2	5	Average of high and low values is used for calculations.
HLC3	6	Average of high, low, and close values is used for calculations.
HLOC4	7	Average of high, low, open, and close values is used for calculations.

FinancialChartType Enum

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance

Specifies the type of financial chart.

Members

Name	Value	Description
Column	0	Shows vertical bars and allows you to compare values of items across categories.
Scatter	1	Uses X and Y coordinates to show patterns within the data.
Line	2	Shows trends over a period of time or across categories.
LineSymbols	3	Shows line chart with a symbol on each data point.
Area	4	Shows line chart with area below the line filled with color.
Candlestick	5	Presents items with high, low, open, and close values. The size of the wick line is determined by the High and Low values, while the size of the bar is determined by the Open and Close values. The bar is displayed using different colors, depending on whether the close value is higher or lower than the open value. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
HighLowOpenClose	6	Displays the same information as a candlestick chart, except that opening values are displayed using lines to the left, while lines to the right indicate closing values. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
HeikinAshi	7	Derived from the candlestick chart and uses information from the current and prior period in order to filter out the noise. These charts cannot be combined with any other series objects. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
LineBreak	8	Filters out noise by focusing exclusively on price changes. These charts cannot be combined with any other series objects. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
Renko	9	Ignores time and focuses on price changes that meet a specified amount. These charts cannot be combined with any other series objects. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
Kagi	10	Ignores time and focuses on price action. These charts cannot be combined with any other series objects. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".
ColumnVolume	11	Identical to the standard Column chart, except that the width of each bar is determined by the Volume value. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "yProperty, volumeProperty". This chart type can only be used at the **`FinancialChart`** level, and should not be applied on **`FinancialSeries`** objects. Only one set of volume data is currently supported per **`FinancialChart`**.
EquiVolume	12	Similar to the Candlestick chart, but shows the high and low values only. In addition, the width of each bar is determined by Volume value. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty, volumeProperty". This chart type can only be used at the **`FinancialChart`** level, and should not be applied on **`FinancialSeries`** objects. Only one set of volume data is currently supported per **`FinancialChart`**.
CandleVolume	13	Identical to the standard Candlestick chart, except that the width of each bar is determined by Volume value. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty, volumeProperty". This chart type can only be used at the **`FinancialChart`** level, and should not be applied on **`FinancialSeries`** objects. Only one set of volume data is currently supported per **`FinancialChart`**.
ArmsCandleVolume	14	Created by Richard Arms, this chart is a combination of EquiVolume and CandleVolume chart types. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty, volumeProperty". This chart type can only be used at the **`FinancialChart`** level, and should not be applied on **`FinancialSeries`** objects. Only one set of volume data is currently supported per **`FinancialChart`**.

Name	Value	Description
PointAndFigure	15	Point and figure financial chart. The data for this chart type can be defined using the **`FinancialChart`** or **`FinancialSeries`** **binding** property as a comma separated value in the following format: "highProperty, lowProperty, closeProperty". This chart type can only be used at the **`FinancialChart`** level, and should not be applied on **`FinancialSeries`** objects.

PointAndFigureScaling Enum

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance

Specifies the scaling mode for point and figure chart.

Members

Name	Value	Description
Traditional	0	Traditional scaling. The box size is calculated automatically based on price range.
Fixed	1	Fixed scaling. The box size is defined by boxSize property.
Dynamic	2	Dynamic(ATR) scaling. The box size is calculated based on ATR.

RangeMode Enum

File
`wijmo.chart.finance.js`
Module
wijmo.chart.finance

Specifies the unit for Kagi and Renko chart types.

Members

Name	Value	Description
Fixed	0	Uses a fixed, positive number for the Kagi chart's reversal amount or Renko chart's box size.
ATR	1	Uses the current Average True Range value for Kagi chart's reversal amount or Renko chart's box size. When ATR is used, the reversal amount or box size option of these charts must be an integer and will be used as the period for the ATR calculation.
Percentage	2	Uses a percentage for the Kagi chart's reversal amount. RangeMode.Percentage is currently not supported with Renko chart types.

wijmo.chart.finance.analytics Module

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics

Analytics extensions for **FinancialChart**.

Classes

- ATR
- BollingerBands
- CCI
- Envelopes
- Fibonacci
- FibonacciArcs
- FibonacciFans
- FibonacciTimeZones
- Macd
- MacdBase
- MacdHistogram
- OverlayIndicatorBase
- RSI
- SingleOverlayIndicatorBase
- Stochastic
- WilliamsR

ATR Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SingleOverlayIndicatorBase
Derived Classes
WjFlexChartAtr

Represents an Average True Range indicator series for the **FinancialChart**.

Average true range is used to measure the volatility of an asset. Average true range does not provide any indication of the price's trend, but rather the degree of price volatility.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): ATR
```

Initializes a new instance of the **ATR** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
ATR

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`SingleOverlayIndicatorBase`
Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

BollingerBands Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
OverlayIndicatorBase
Derived Classes
WjFlexChartBollingerBands

Represents a Bollinger Bands® overlay series for the **FinancialChart**.

Bollinger Bands is a registered trademark of John Bollinger.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- multiplier
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): BollingerBands
```

Initializes a new instance of the **BollingerBands** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
BollingerBands

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

multiplier

Gets or sets the standard deviation multiplier.

Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

CCI Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SingleOverlayIndicatorBase
Derived Classes
WjFlexChartCci

Represents a Commodity Channel Index indicator series for the **FinancialChart**.

The commodity channel index is an oscillator that measures an asset's current price level relative to an average price level over a specified period of time.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- constant
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): CCI
```

Initializes a new instance of the **CCI** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
CCI

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

constant

Gets or sets the constant value for the CCI calculation. The default value is 0.015.

Type
number

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`SingleOverlayIndicatorBase`
Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

Envelopes Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
OverlayIndicatorBase
Derived Classes
WjFlexChartEnvelopes

Represents a Moving Average Envelopes overlay series for the **FinancialChart**.

Moving average envelopes are moving averages set above and below a standard moving average. The amount above/below the standard moving average is percentage based and dictated by the **size** property.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- size
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Envelopes
```

Initializes a new instance of the **Envelopes** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Envelopes

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Type
any

size

Gets or set the size of the moving average envelopes. The default value is 2.5 percent (0.025).

Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

type

Gets or sets the moving average type for the envelopes. The default value is Simple.

Type
`MovingAverageType`

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

Fibonacci Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SeriesBase
Derived Classes
WjFlexChartFibonacci

Represents a Fibonacci Retracements tool for the **FinancialChart**. The tool enables the calculation and plotting of various alert levels that are useful in financial charts.

To add Fibonacci tool to a **FinancialChart** control, create an instance of the **Fibonacci** and add it to the **series** collection of the chart. For example:

```
// create chart
var chart = new wijmo.chart.finance.FinancialChart('#chartElement');

// create Fibonacci tool
var ftool = new wijmo.chart.finance.analytics.Fibonacci();
chart.series.push(ftool);
```

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- high
- hostElement
- itemsSource
- labelPosition
- legendElement
- levels
- low
- maxX
- minX
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- uptrend
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Fibonacci
```

Initializes a new instance of the **Fibonacci** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Fibonacci

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

high

Gets or sets the high value of **Fibonacci** tool.

If not specified, the high value is caclulated based on data values provided by the **itemsSource**.

Type
number

hostElement

Gets the series host element.

Inherited From
SeriesBase
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

labelPosition

Gets or sets the label position for levels in **`Fibonacci`** tool.

Type
`LabelPosition`

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [0, 23.6, 38.2, 50, 61.8, 100].

Type
number[]

low

Gets or sets the low value of **`Fibonacci`** tool.

If not specified, the low value is calculated based on data values provided by **itemsSource**.

Type
number

maxX

Gets or sets the x maximum value of the **`Fibonacci`** tool.

If not specified, current maximum of x-axis is used. The value can be specified as a number or Date object.

Type
any

minX

Gets or sets the x minimal value of the **`Fibonacci`** tool.

If not specified, current minimum of x-axis is used. The value can be specified as a number or Date object.

Type
any

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

uptrend

Gets or sets a value indicating whether to create uptrending **`Fibonacci`** tool.

Default value is true(uptrend). If the value is false, the downtrending levels are plotted.

Type
boolean

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

FibonacciArcs Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SeriesBase
Derived Classes
WjFlexChartFibonacciArcs

Represents a Fibonacci Arcs tool for the **FinancialChart**.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- itemsSource
- labelPosition
- legendElement
- levels
- name
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FibonacciArcs
```

Initializes a new instance of the **FibonacciArcs** class.

Parameters

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data.

Returns
FibonacciArcs

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

end

Gets or sets the ending **`DataPoint`** for the base line.

The **`DataPoint`** x value can be a number or a Date object (for time-based data).

Unlike some of the other Fibonacci tools, the ending **`DataPoint`** is **not** calculated automatically if undefined.

Type
`DataPoint`

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **ICollectionView** object that contains the series data.

Inherited From
SeriesBase
Type
any

labelPosition

Gets or sets the **LabelPosition** for levels in **FibonacciArcs** tool.

Type
LabelPosition

legendElement

Gets the series element in the legend.

Inherited From
SeriesBase
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [38.2, 50, 61.8].

Type
number[]

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

start

Gets or sets the starting `DataPoint` for the base line.

The `DataPoint` x value can be a number or a Date object (for time-based data).

Unlike some of the other Fibonacci tools, the starting `DataPoint` is **not** calculated automatically if undefined.

Type
`DataPoint`

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

FibonacciFans Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SeriesBase
Derived Classes
WjFlexChartFibonacciFans

Represents a Fibonacci Fans tool for the **FinancialChart**.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- itemsSource
- labelPosition
- legendElement
- levels
- name
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FibonacciFans
```

Initializes a new instance of the **FibonacciFans** class.

Parameters

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data.

Returns
FibonacciFans

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

end

Gets or sets the ending **`DataPoint`** for the base line.

If not set, the starting **`DataPoint`** is calculated automatically. The **`DataPoint`** x value can be a number or a Date object (for time-based data).

Type
`DataPoint`

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **ICollectionView** object that contains the series data.

Inherited From
SeriesBase
Type
any

labelPosition

Gets or sets the **LabelPosition** for levels in **FibonacciFans** tool.

Type
LabelPosition

legendElement

Gets the series element in the legend.

Inherited From
SeriesBase
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [0, 23.6, 38.2, 50, 61.8, 100].

Type
number[]

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

start

Gets or sets the starting `DataPoint` for the base line.

If not set, the starting `DataPoint` is calculated automatically. The `DataPoint` x value can be a number or a Date object (for time-based data).

Type
`DataPoint`

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

FibonacciTimeZones Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SeriesBase
Derived Classes
WjFlexChartFibonacciTimeZones

Represents a Fibonacci Time Zones tool for the **FinancialChart**.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- endX
- hostElement
- itemsSource
- labelPosition
- legendElement
- levels
- name
- startX
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FibonacciTimeZones
```

Initializes a new instance of the **FibonacciTimeZones** class.

Parameters

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data.

Returns
FibonacciTimeZones

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

endX

Gets or sets the ending X data point for the time zones.

If not set, the ending X data point is calculated automatically. The value can be a number or a Date object (for time-based data).

Type
any

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **ICollectionView** object that contains the series data.

Inherited From
SeriesBase
Type
any

labelPosition

Gets or sets the **LabelPosition** for levels in **FibonacciTimeZones** tool.

Type
LabelPosition

legendElement

Gets the series element in the legend.

Inherited From
SeriesBase
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [0, 1, 2, 3, 5, 8, 13, 21, 34].

Type
number[]

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

startX

Gets or sets the starting X data point for the time zones.

If not set, the starting X data point is calculated automatically. The value can be a number or a Date object (for time-based data).

Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

Macd Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
MacdBase
Derived Classes
WjFlexChartMacd

Represents a Moving Average Convergence/Divergence (MACD) indicator series for the **FinancialChart**.

The MACD indicator is designed to reveal changes in strength, direction, momentum, and duration of an asset's price trend.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- itemsSource
- legendElement
- name
- slowPeriod
- smoothingPeriod
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Macd
```

Initializes a new instance of the **Macd** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Macd

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

fastPeriod

Gets or sets the fast exponential moving average period for the MACD line.

Inherited From
`MacdBase`
Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

slowPeriod

Gets or sets the slow exponential moving average period for the MACD line.

Inherited From
`MacdBase`
Type
number

smoothingPeriod

Gets or sets the exponential moving average period for the signal line.

Inherited From
`MacdBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

styles

Gets or sets the styles for the MACD and Signal lines.

The following options are supported:

```
series.styles = {
  macdLine: {
     stroke: 'red',
     strokeWidth: 1
  },
  signalLine: {
     stroke: 'green',
     strokeWidth: 1
  },
}
```

Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

MacdBase Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
OverlayIndicatorBase
Derived Classes
Macd, **MacdHistogram**

Base class for **Macd** and **MacdHistogram** series (abstract).

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- itemsSource
- legendElement
- name
- slowPeriod
- smoothingPeriod
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): MacdBase
```

Initializes a new instance of the **MacdBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
MacdBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

fastPeriod

Gets or sets the fast exponential moving average period for the MACD line.

Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

slowPeriod

Gets or sets the slow exponential moving average period for the MACD line.

Type
number

smoothingPeriod

Gets or sets the exponential moving average period for the signal line.

Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

MacdHistogram Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
MacdBase
Derived Classes
WjFlexChartMacdHistogram

Represents a Moving Average Convergence/Divergence (MACD) Histogram indicator series for the **FinancialChart**.

The MACD indicator is designed to reveal changes in strength, direction, momentum, and duration of an asset's price trend.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- fastPeriod
- hostElement
- itemsSource
- legendElement
- name
- slowPeriod
- smoothingPeriod
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): MacdHistogram
```

Initializes a new instance of the **MacdHistogram** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
MacdHistogram

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

fastPeriod

Gets or sets the fast exponential moving average period for the MACD line.

Inherited From
`MacdBase`
Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

slowPeriod

Gets or sets the slow exponential moving average period for the MACD line.

Inherited From
`MacdBase`
Type
number

smoothingPeriod

Gets or sets the exponential moving average period for the signal line.

Inherited From
`MacdBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

OverlayIndicatorBase Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SeriesBase
Derived Classes
BollingerBands, **Envelopes**, **MacdBase**, **SingleOverlayIndicatorBase**, **Stochastic**

Base class for overlay and indicator series (abstract).

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): OverlayIndicatorBase
```

Initializes a new instance of the **OverlayIndicatorBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
OverlayIndicatorBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

RSI Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SingleOverlayIndicatorBase
Derived Classes
WjFlexChartRsi

Represents a Relative Strength Index indicator series for the **FinancialChart**.

Relative strength index is a momentum oscillator designed to measure the current and historical strength or weakness of an asset based on the closing prices of a recent trading period.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): RSI
```

Initializes a new instance of the **RSI** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
RSI

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`SingleOverlayIndicatorBase`
Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

SingleOverlayIndicatorBase Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
OverlayIndicatorBase
Derived Classes
ATR, **CCI**, **RSI**, **WilliamsR**

Base class for overlay and indicator series that render a single series (abstract).

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SingleOverlayIndicatorBase
```

Initializes a new instance of the **SingleOverlayIndicatorBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
SingleOverlayIndicatorBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

Stochastic Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
OverlayIndicatorBase
Derived Classes
WjFlexChartStochastic

Represents a Stochastic Oscillator indicator series for the **FinancialChart**.

Stochastic oscillators are momentum indicators designed to predict price turning points by comparing an asset's closing price to its high-low range.

The **Stochastic** series can be used for fast (default), slow and full stochastic oscillators. To create a slow or full stochastic oscillator, set the **smoothingPeriod** to an integer value greater than one; slow stochastic oscillators generally use a fixed **smoothingPeriod** of three. To create or revert to a fast stochastic oscillator, set the **smoothingPeriod** to an integer value of one.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- dPeriod
- hostElement
- itemsSource
- kPeriod
- legendElement
- name
- smoothingPeriod
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Stochastic
```

Initializes a new instance of the **`Stochastic`** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
Stochastic

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
`SeriesBase`
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the `ICollectionView` object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

dPeriod

Gets or sets the period for the %D simple moving average.

Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

kPeriod

Gets or sets the period for the %K calculation.

Type
number

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

smoothingPeriod

Gets or sets the smoothing period for full %K.

Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

styles

Gets or sets the styles for the %K and %D lines.

The following options are supported:

```
series.styles = {
  kLine: {
     stroke: 'red',
     strokeWidth: 1
  },
  dLine: {
     stroke: 'green',
     strokeWidth: 1
  },
}
```

Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WilliamsR Class

File
`wijmo.chart.finance.analytics.js`
Module
wijmo.chart.finance.analytics
Base Class
SingleOverlayIndicatorBase
Derived Classes
WjFlexChartWilliamsR

Represents a Willaims %R indicator series for the **FinancialChart**.

Williams %R is a momentum indicator that is the inverse of a fast stochastic oscillator (**Stochastic**). The Williams %R indicator is designed to tell whether an asset is trading near the high or low of its trading range.

Constructor

- constructor

Properties

- altStyle
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- itemsSource
- legendElement
- name
- period
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility

Methods

- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): WilliamsR
```

Initializes a new instance of the **WilliamsR** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Returns
WilliamsR

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

axisX

Gets or sets the x-axis for the series.

Inherited From
SeriesBase
Type
Axis

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the `FlexChart` object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`SingleOverlayIndicatorBase`
Type
any

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

Methods

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

wijmo.olap Module

File
`wijmo.olap.js`
Module
wijmo.olap

Contains components that provide OLAP functionality such as pivot tables and charts.

The **PivotEngine** class is responsible for summarizing raw data into pivot views.

The **PivotPanel** control provides a UI for editing the pivot views by dragging fields into view lists and editing their properties.

The **PivotGrid** control extends the **FlexGrid** to display pivot tables with collapsible row and column groups.

The **PivotChart** control provides visual representations of pivot tables with hierarchical axes.

Classes

- CubePivotField
- DetailDialog
- PivotChart
- PivotCollectionView
- PivotEngine
- PivotField
- PivotFieldCollection
- PivotFieldEditor
- PivotFilter
- PivotFilterEditor
- PivotGrid
- PivotPanel
- ProgressEventArgs

Enums

- DimensionType
- LegendVisibility
- PivotChartType
- ShowAs
- ShowTotals

CubePivotField Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
PivotField

Extends the **PivotField** class to represent a field in a server-based cube data source.

Constructor

- constructor

Properties

- aggregate
- binding
- collectionView
- dataType
- descending
- dimensionType
- engine
- filter
- format
- header
- isActive
- isContentHtml
- key
- parentField
- showAs
- sortComparer
- subFields
- weightField
- width
- wordWrap

Methods

- onPropertyChanged

Events

- propertyChanged

Constructor

constructor

```
constructor(engine: PivotEngine, binding: string, header?: string, options?: any): CubePivotField
```

Initializes a new instance of the **PivotField** class.

Parameters

- **engine**: **PivotEngine**
  PivotEngine that owns this field.
- **binding**: **string**
  Property that this field is bound to.
- **header**: **string** OPTIONAL
  Header shown to identify this field (defaults to the binding).
- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the field.

Returns
CubePivotField

Properties

aggregate

Gets or sets how the field should be summarized.

Inherited From
PivotField
Type
Aggregate

binding

Gets or sets the name of the property the field is bound to.

Inherited From
PivotField
Type
string

collectionView

Gets the **`ICollectionView`** bound to this field.

Inherited From
`PivotField`
Type
`ICollectionView`

dataType

Gets or sets the data type of the field.

Inherited From
`PivotField`
Type
`DataType`

descending

Gets or sets a value that determines whether keys should be sorted in descending order for this field.

Inherited From
`PivotField`
Type
boolean

dimensionType

Gets or sets the dimension type of the field.

Type
`DimensionType`

engine

Gets a reference to the **PivotEngine** that owns this **PivotField**.

Inherited From
PivotField
Type
PivotEngine

filter

Gets a reference to the **PivotFilter** used to filter values for this field.

Inherited From
PivotField
Type
PivotFilter

format

Gets or sets the format to use when displaying field values.

Inherited From
PivotField
Type
string

header

Gets or sets a string used to represent this field in the user interface.

Type
string

isActive

Gets or sets a value that determines whether this field is currently being used in the view.

Setting this property to true causes the field to be added to the view's **`rowFields`** or **`valueFields`**, depending on the field's data type.

Inherited From
`PivotField`
Type
boolean

isContentHtml

Gets or sets a value indicating whether items in this field contain HTML content rather than plain text.

Inherited From
`PivotField`
Type
boolean

key

Gets the key for this **`CubePivotField`**.

For this type of field, the key is the field's **`binding`**.

Type
string

parentField

Gets this field's parent field.

When you drag the same field into the Values list multiple times, copies of the field are created so you can use the same binding with different parameters. The copies keep a reference to their parent fields.

Inherited From
`PivotField`
Type
`PivotField`

showAs

Gets or sets how the field results should be formatted.

Inherited From
`PivotField`
Type
`ShowAs`

sortComparer

Gets or sets a function used to compare values when sorting.

If provided, the sort comparer function should take as parameters two values of any type, and should return -1, 0, or +1 to indicate whether the first value is smaller than, equal to, or greater than the second. If the sort comparer returns null, the standard built-in comparer is used.

This **`sortComparer`** property allows you to use custom comparison algorithms that in some cases result in sorting sequences that are more consistent with user's expectations than plain string comparisons.

The example below shows a typical use for the **`sortComparer`** property:

```
// define list of products
app.products = 'Wijmo,Aoba,Olap,Xuni'.split(',');

// sort products by position in the 'app.products' array
ng.viewDefinitionChanged.addHandler(function () {
  var fld = ng.fields.getField('Product');
  if (fld) {
    fld.sortComparer = function (val1, val2) {
      return app.products.indexOf(val1) - app.products.indexOf(val2);
    }
  }
});
```

Inherited From
`PivotField`
Type
`Function`

subFields

Gets this field's child fields.

Type
`CubePivotField`[]

weightField

Gets or sets the **`PivotField`** used as a weight for calculating aggregates on this field.

If this property is set to null, all values are assumed to have weight one.

This property allows you to calculate weighted averages and totals. For example, if the data contains a 'Quantity' field and a 'Price' field, you could use the 'Price' field as a value field and the 'Quantity' field as a weight. The output would contain a weighted average of the data.

Inherited From
`PivotField`
Type
`PivotField`

width

Gets or sets the preferred width to be used for showing this field in the user interface.

Inherited From
`PivotField`
Type
number

wordWrap

Gets or sets a value that indicates whether the content of this field should be allowed to wrap within cells.

Inherited From
`PivotField`
Type
boolean

Methods

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

Raises the **propertyChanged** event.

Parameters

- **e**: **PropertyChangedEventArgs**
  `PropertyChangedEventArgs` that contains the property name, old, and new values.

Inherited From
`PivotField`
Returns
void

Events

propertyChanged

Occurs when the value of a property in this **Range** changes.

Inherited From
`PivotField`
Arguments
`PropertyChangedEventArgs`

DetailDialog Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
Control

Represents a dialog used to display details for a grid cell.

Constructor

- constructor

Properties

- controlTemplate
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, options?): DetailDialog
```

Initializes a new instance of the **DetailDialog** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
DetailDialog

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **PivotFieldEditor** controls.

Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

PivotChart Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
Control
Derived Classes
WjPivotChart

Provides visual representations of **wijmo.olap** pivot tables.

To use the control, set its **itemsSource** property to an instance of a **PivotPanel** control or to a **PivotEngine**.

Constructor

- constructor

Properties

- chartType
- engine
- flexChart
- flexPie
- hostElement
- isDisabled
- isTouching
- isUpdating
- itemsSource
- legendPosition
- maxPoints
- maxSeries
- rightToLeft
- showHierarchicalAxes
- showLegend
- showTitle
- showTotals
- stacking

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, options?): PivotChart
```

Initializes a new instance of the **PivotChart** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
PivotChart

Properties

chartType

Gets or sets the type of chart to create.

Type
PivotChartType

engine

Gets a reference to the **PivotEngine** that owns this **PivotChart**.

Type
PivotEngine

flexChart

Gets a reference to the inner **FlexChart** control.

Type
FlexChart

flexPie

Gets a reference to the inner **FlexPie** control.

Type
`FlexPie`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemsSource

Gets or sets the `PivotEngine` or `PivotPanel` that provides data for this `PivotChart`.

Type
any

legendPosition

Gets or sets a value that determines whether and where the legend appears in relation to the plot area.

Type
`Position`

maxPoints

Gets or sets the maximum number of points to be shown in each series.

Type
number

maxSeries

Gets or sets the maximum number of data series to be shown in the chart.

Type
number

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showHierarchicalAxes

Gets or sets a value that determines whether the chart should group axis annotations for grouped data.

Type
boolean

showLegend

Gets or sets a value that determines whether the chart should include a legend.

Type
`LegendVisibility`

showTitle

Gets or sets a value that determines whether the chart should include a title.

Type
boolean

showTotals

Gets or sets a value that determines whether the chart should include only totals.

Type
boolean

stacking

Gets or sets a value that determines whether and how the series objects are stacked.

Type
`Stacking`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

PivotCollectionView Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
CollectionView

Extends the **CollectionView** class to preserve the position of subtotal rows when sorting.

Constructor

- constructor

Properties

- canAddNew
- canCancelEdit
- canChangePage
- canFilter
- canGroup
- canRemove
- canSort
- currentAddItem
- currentEditItem
- currentItem
- currentPosition
- engine
- filter
- getError
- groupDescriptions
- groups
- isAddingNew
- isEditingItem
- isEmpty
- isPageChanging
- isUpdating
- itemCount
- items
- itemsAdded
- itemsEdited
- itemsRemoved
- newItemCreator
- pageCount
- pageIndex
- pageSize
- sortComparer
- sortConverter
- sortDescriptions
- sourceCollection
- totalItemCount
- trackChanges
- useStableSort

Methods

- addNew
- beginUpdate
- cancelEdit
- cancelNew
- clearChanges
- commitEdit
- commitNew
- contains
- deferUpdate
- editItem
- endUpdate
- getAggregate
- implementsInterface
- moveCurrentTo
- moveCurrentToFirst
- moveCurrentToLast
- moveCurrentToNext
- moveCurrentToPosition
- moveCurrentToPrevious
- moveToFirstPage
- moveToLastPage
- moveToNextPage
- moveToPage
- moveToPreviousPage
- onCollectionChanged
- onCurrentChanged
- onCurrentChanging
- onPageChanged
- onPageChanging
- onSourceCollectionChanged
- onSourceCollectionChanging
- refresh
- remove
- removeAt

Events

- collectionChanged
- currentChanged
- currentChanging

- pageChanged
- pageChanging
- sourceCollectionChanged
- sourceCollectionChanging

Constructor

constructor

```
constructor(engine: PivotEngine): PivotCollectionView
```

Initializes a new instance of the **`PivotCollectionView`** class.

Parameters

- **engine**: **PivotEngine**
  `PivotEngine` that owns this collection.

Returns
PivotCollectionView

Properties

canAddNew

Gets a value that indicates whether a new item can be added to the collection.

Inherited From
`CollectionView`
Type
boolean

canCancelEdit

Gets a value that indicates whether the collection view can discard pending changes and restore the original values of an edited object.

Inherited From
`CollectionView`
Type
boolean

canChangePage

Gets a value that indicates whether the **`pageIndex`** value can change.

Inherited From
`CollectionView`
Type
boolean

canFilter

Gets a value that indicates whether this view supports filtering via the **`filter`** property.

Inherited From
`CollectionView`
Type
boolean

canGroup

Gets a value that indicates whether this view supports grouping via the **`groupDescriptions`** property.

Inherited From
`CollectionView`
Type
boolean

canRemove

Gets a value that indicates whether items can be removed from the collection.

Inherited From
`CollectionView`
Type
boolean

canSort

Gets a value that indicates whether this view supports sorting via the **`sortDescriptions`** property.

Inherited From
`CollectionView`
Type
boolean

currentAddItem

Gets the item that is being added during the current add transaction.

Inherited From
`CollectionView`
Type
any

currentEditItem

Gets the item that is being edited during the current edit transaction.

Inherited From
`CollectionView`
Type
any

currentItem

Gets or sets the current item in the view.

Inherited From
`CollectionView`
Type
any

currentPosition

Gets the ordinal position of the current item in the view.

Inherited From
`CollectionView`
Type
number

engine

Gets a reference to the **`PivotEngine`** that owns this view.

Type
`PivotEngine`

filter

Gets or sets a callback used to determine if an item is suitable for inclusion in the view.

The callback function should return true if the item passed in as a parameter should be included in the view.

NOTE: If the filter function needs a scope (i.e. a meaningful 'this' value) remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
collectionView.filter = this._filter.bind(this);
```

Inherited From
`CollectionView`
Type
`IPredicate`

getError

Gets or sets a callback that determines whether a specific property of an item contains validation errors.

If provided, the callback should take two parameters containing the item and the property to validate, and should return a string describing the error (or null if there are no errors).

For example:

```
var view = new wijmo.collections.CollectionView(data, {
    getError: function (item, property) {
        switch (property) {
            case 'country':
                return countries.indexOf(item.country) < 0
                    ? 'Invalid Country'
                    : null;
            case 'downloads':
            case 'sales':
            case 'expenses':
                return item[property] < 0
                    ? 'Cannot be negative!'
                    : null;
            case 'active':
                return item.active && item.country.match(/US|UK/)
                    ? 'No active items allowed in the US or UK!'
                    : null;
        }
        return null;
    }
});
```

Inherited From
`CollectionView`
Type
Function

groupDescriptions

Gets a collection of **`GroupDescription`** objects that describe how the items in the collection are grouped in the view.

Inherited From
`CollectionView`
Type
`ObservableArray`

groups

Gets an array of **`CollectionViewGroup`** objects that represents the top-level groups.

Inherited From
`CollectionView`
Type
`CollectionViewGroup`[]

isAddingNew

Gets a value that indicates whether an add transaction is in progress.

Inherited From
`CollectionView`
Type
boolean

isEditingItem

Gets a value that indicates whether an edit transaction is in progress.

Inherited From
`CollectionView`
Type
boolean

isEmpty

Gets a value that indicates whether this view contains no items.

Inherited From
`CollectionView`
Type
boolean

isPageChanging

Gets a value that indicates whether the page index is changing.

Inherited From
`CollectionView`
Type
boolean

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **`beginUpdate`** and **`endUpdate`**).

Inherited From
`CollectionView`
Type

itemCount

Gets the total number of items in the view taking paging into account.

Inherited From
`CollectionView`
Type
number

items

Gets items in the view.

Inherited From
`CollectionView`
Type
any[]

itemsAdded

Gets an **`ObservableArray`** containing the records that were added to the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

itemsEdited

Gets an **`ObservableArray`** containing the records that were edited in the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

itemsRemoved

Gets an **`ObservableArray`** containing the records that were removed from the collection since **`trackChanges`** was enabled.

Inherited From
`CollectionView`
Type
`ObservableArray`

newItemCreator

Gets or sets a function that creates new items for the collection.

If the creator function is not supplied, the **`CollectionView`** will try to create an uninitialized item of the appropriate type.

If the creator function is supplied, it should be a function that takes no parameters and returns an initialized object of the proper type for the collection.

Inherited From
`CollectionView`
Type
Function

pageCount

Gets the total number of pages.

Inherited From
`CollectionView`
Type
number

pageIndex

Gets the zero-based index of the current page.

Inherited From
`CollectionView`
Type
number

pageSize

Gets or sets the number of items to display on a page.

Inherited From
`CollectionView`
Type
number

sortComparer

Gets or sets a function used to compare values when sorting.

If provided, the sort comparer function should take as parameters two values of any type, and should return -1, 0, or +1 to indicate whether the first value is smaller than, equal to, or greater than the second. If the sort comparer returns null, the standard built-in comparer is used.

This **`sortComparer`** property allows you to use custom comparison algorithms that in some cases result in sorting sequences that are more consistent with user's expectations than plain string comparisons.

For example, see **Dave Koele's Alphanum algorithm**. It breaks up strings into chunks composed of strings or numbers, then sorts number chunks in value order and string chunks in ASCII order. Dave calls the result a "natural sorting order".

The example below shows a typical use for the **`sortComparer`** property:

```
// create a CollectionView with a custom sort comparer
var dataCustomSort = new wijmo.collections.CollectionView(data, {
  sortComparer: function (a, b) {
    return wijmo.isString(a) && wijmo.isString(b)
      ? alphanum(a, b) // custom comparer used for strings
      : null; // use default comparer used for everything else
  }
});
```

Inherited From
`CollectionView`
Type
Function

sortConverter

Gets or sets a function used to convert values when sorting.

If provided, the function should take as parameters a **SortDescription**, a data item, and a value to convert, and should return the converted value.

This property provides a way to customize sorting. For example, the **FlexGrid** control uses it to sort mapped columns by display value instead of by raw value.

For example, the code below causes a **CollectionView** to sort the 'country' property, which contains country code integers, using the corresponding country names:

```
var countries = 'US,Germany,UK,Japan,Italy,Greece'.split(',');
collectionView.sortConverter = function (sd, item, value) {
  if (sd.property == 'countryMapped') {
    value = countries[value]; // convert country id into name
  }
  return value;
}
```

Inherited From
CollectionView
Type
Function

sortDescriptions

Gets a collection of **SortDescription** objects that describe how the items in the collection are sorted in the view.

Inherited From
CollectionView
Type
ObservableArray

sourceCollection

Gets or sets the underlying (unfiltered and unsorted) collection.

Inherited From
CollectionView
Type
any

totalItemCount

Gets the total number of items in the view before paging is applied.

Inherited From
`CollectionView`
Type
number

trackChanges

Gets or sets a value that determines whether the control should track changes to the data.

If **`trackChanges`** is set to true, the **`CollectionView`** keeps track of changes to the data and exposes them through the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

Tracking changes is useful in situations where you need to update the server after the user has confirmed that the modifications are valid.

After committing or cancelling changes, use the **`clearChanges`** method to clear the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

The **`CollectionView`** only tracks changes made when the proper **`CollectionView`** methods are used (**`editItem`**/**`commitEdit`**, **`addNew`**/**`commitNew`**, and **`remove`**). Changes made directly to the data are not tracked.

Inherited From
`CollectionView`
Type
boolean

useStableSort

Gets or sets whether to use a stable sort algorithm.

Stable sorting algorithms maintain the relative order of records with equal keys. For example, consider a collection of objects with an "Amount" field. If you sort the collection by "Amount", a stable sort will keep the original order of records with the same Amount value.

This property is false by default, which causes the **`CollectionView`** to use JavaScript's built-in sort method, which is very fast but not stable. Setting the **`useStableSort`** property to true increases sort times by 30% to 50%, which can be significant for large collections.

Inherited From
`CollectionView`
Type
boolean

Methods

addNew

```
addNew(): any
```

Creates a new item and adds it to the collection.

This method takes no parameters. It creates a new item, adds it to the collection, and defers refresh operations until the new item is committed using the **commitNew** method or canceled using the **cancelNew** method.

The code below shows how the **addNew** method is typically used:

```
// create the new item, add it to the collection
var newItem = view.addNew();

// initialize the new item
newItem.id = getFreshId();
newItem.name = 'New Customer';

// commit the new item so the view can be refreshed
view.commitNew();
```

You can also add new items by pushing them into the **sourceCollection** and then calling the **refresh** method. The main advantage of **addNew** is in user-interactive scenarios (like adding new items in a data grid), because it gives users the ability to cancel the add operation. It also prevents the new item from being sorted or filtered out of view until the add operation is committed.

Inherited From
CollectionView
Returns
any

beginUpdate

```
beginUpdate(): void
```

Suspend refreshes until the next call to **endUpdate**.

Inherited From
CollectionView
Returns
void

cancelEdit

```
cancelEdit(): void
```

Ends the current edit transaction and, if possible, restores the original value to the item.

Inherited From
`CollectionView`
Returns
void

cancelNew

```
cancelNew(): void
```

Ends the current add transaction and discards the pending new item.

Inherited From
`CollectionView`
Returns
void

clearChanges

```
clearChanges(): void
```

Clears all changes by removing all items in the **`itemsAdded`**, **`itemsRemoved`**, and **`itemsEdited`** collections.

Call this method after committing changes to the server or after refreshing the data from the server.

Inherited From
`CollectionView`
Returns
void

commitEdit

```
commitEdit(): void
```

Ends the current edit transaction and saves the pending changes.

Inherited From
`CollectionView`
Returns
void

commitNew

```
commitNew(): void
```

Ends the current add transaction and saves the pending new item.

Inherited From
`CollectionView`
Returns
void

contains

```
contains(item: any): boolean
```

Returns a value indicating whether a given item belongs to this view.

Parameters

- **item**: **any**
  Item to seek.

Inherited From
`CollectionView`
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The collection will not be refreshed until the function finishes. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
`CollectionView`
Returns
void

editItem

```
editItem(item: any): void
```

Begins an edit transaction of the specified item.

Parameters

- **item**: **any**
  Item to be edited.

Inherited From
`CollectionView`
Returns
void

endUpdate

```
endUpdate(): void
```

Resume refreshes suspended by a call to **beginUpdate**.

Inherited From
CollectionView
Returns
void

getAggregate

```
getAggregate(aggType: Aggregate, binding: string, currentPage?: boolean): void
```

Calculates an aggregate value for the items in this collection.

Parameters

- **aggType**: **Aggregate**
  Type of aggregate to calculate.
- **binding**: **string**
  Property to aggregate on.
- **currentPage**: **boolean** OPTIONAL
  Whether to include only items on the current page.

Inherited From
CollectionView
Returns
void

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
CollectionView
Returns
boolean

moveCurrentTo

```
moveCurrentTo(item: any): boolean
```

Sets the specified item to be the current item in the view.

Parameters

- **item**: **any**
  Item that will become current.

Inherited From
CollectionView
Returns
boolean

moveCurrentToFirst

```
moveCurrentToFirst(): boolean
```

Sets the first item in the view as the current item.

Inherited From
CollectionView
Returns
boolean

moveCurrentToLast

```
moveCurrentToLast(): boolean
```

Sets the last item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToNext

```
moveCurrentToNext(): boolean
```

Sets the item after the current item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToPosition

```
moveCurrentToPosition(index: number): boolean
```

Sets the item at the specified index in the view as the current item.

Parameters

- **index**: **number**
  Index of the item that will become current.

Inherited From
`CollectionView`
Returns
boolean

moveCurrentToPrevious

```
moveCurrentToPrevious(): boolean
```

Sets the item before the current item in the view as the current item.

Inherited From
`CollectionView`
Returns
boolean

moveToFirstPage

```
moveToFirstPage(): boolean
```

Sets the first page as the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToLastPage

```
moveToLastPage(): boolean
```

Sets the last page as the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToNextPage

```
moveToNextPage(): boolean
```

Moves to the page after the current page.

Inherited From
`CollectionView`
Returns
boolean

moveToPage

```
moveToPage(index: number): boolean
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index of the page to move to.

Inherited From
`CollectionView`
Returns
boolean

moveToPreviousPage

```
moveToPreviousPage(): boolean
```

Moves to the page before the current page.

Inherited From
`CollectionView`
Returns
boolean

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
CollectionView
Returns
void

onCurrentChanged

```
onCurrentChanged(e?: EventArgs): void
```

Raises the **currentChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
CollectionView
Returns
void

onCurrentChanging

```
onCurrentChanging(e: CancelEventArgs): boolean
```

Raises the **currentChanging** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
CollectionView
Returns
boolean

onPageChanged

```
onPageChanged(e?: EventArgs): void
```

Raises the **pageChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
CollectionView
Returns
void

onPageChanging

```
onPageChanging(e: PageChangingEventArgs): boolean
```

Raises the **pageChanging** event.

Parameters

- **e**: **PageChangingEventArgs**
  `PageChangingEventArgs` that contains the event data.

Inherited From
`CollectionView`
Returns
boolean

onSourceCollectionChanged

```
onSourceCollectionChanged(e?: EventArgs): void
```

Raises the **sourceCollectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`CollectionView`
Returns
void

onSourceCollectionChanging

```
onSourceCollectionChanging(e: CancelEventArgs): boolean
```

Raises the **sourceCollectionChanging** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
CollectionView
Returns
boolean

refresh

```
refresh(): void
```

Re-creates the view using the current sort, filter, and group parameters.

Inherited From
CollectionView
Returns
void

remove

```
remove(item: any): void
```

Removes the specified item from the collection.

Parameters

- **item**: **any**
  Item to be removed from the collection.

Inherited From
CollectionView
Returns
void

removeAt

```
removeAt(index: number): void
```

Removes the item at the specified index from the collection.

Parameters

- **index**: **number**
  Index of the item to be removed from the collection. The index is relative to the view, not to the source collection.

Inherited From
`CollectionView`
Returns
void

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`CollectionView`
Arguments
`NotifyCollectionChangedEventArgs`

currentChanged

Occurs after the current item changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

currentChanging

Occurs before the current item changes.

Inherited From
`CollectionView`
Arguments
`CancelEventArgs`

pageChanged

Occurs after the page index changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

pageChanging

Occurs before the page index changes.

Inherited From
`CollectionView`
Arguments
`PageChangingEventArgs`

sourceCollectionChanged

Occurs after the value of the **`sourceCollection`** property changes.

Inherited From
`CollectionView`
Arguments
`EventArgs`

sourceCollectionChanging

Occurs before the value of the **`sourceCollection`** property changes.

Inherited From
`CollectionView`
Arguments
`CancelEventArgs`

PivotEngine Class

File
`wijmo.olap.js`
Module
wijmo.olap

Provides a user interface for interactively transforming regular data tables into Olap pivot tables.

Tabulates data in the **itemsSource** collection according to lists of fields and creates the **pivotView** collection containing the aggregated data.

Pivot tables group data into one or more dimensions. The dimensions are represented by rows and columns on a grid, and the data is stored in the grid cells.

Constructor

- constructor

Properties

- allowFieldEditing
- async
- autoGenerateFields
- collectionView
- columnFields
- defaultFilterType
- fields
- filterFields
- isUpdating
- isViewDefined
- itemsSource
- pivotView
- rowFields
- serverMaxDetail
- serverPollInterval
- serverTimeout
- showColumnTotals
- showRowTotals
- showZeros
- totalsBeforeData
- valueFields
- viewDefinition

Methods

- beginUpdate
- cancelPendingUpdates
- deferUpdate
- editField
- endUpdate
- getDetail
- getDetailView
- getKeys
- invalidate
- onError
- onItemsSourceChanged
- onUpdatedView
- onUpdatingView
- onViewDefinitionChanged
- refresh
- removeField

Events

- error
- itemsSourceChanged
- updatedView
- updatingView
- viewDefinitionChanged

Constructor

constructor

```
constructor(options?: any): PivotEngine
```

Initializes a new instance of the **PivotEngine** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the field.

Returns
PivotEngine

Properties

allowFieldEditing

Gets or sets a value that determines whether users should be allowed to edit the properties of the **PivotField** objects owned by this **PivotEngine**.

Type
boolean

async

Gets or sets a value that determines whether view updates should be generated asynchronously.

This property is set to true by default, so summaries over large data sets are performed asynchronously to prevent stopping the UI thread.

Type
boolean

autoGenerateFields

Gets or sets a value that determines whether the engine should generate fields automatically based on the **itemsSource**.

Type
boolean

collectionView

Gets the **`ICollectionView`** that contains the raw data.

Type
`ICollectionView`

columnFields

Gets the list of **`PivotField`** objects that define the fields shown as columns in the output table.

Type
`PivotFieldCollection`

defaultFilterType

Gets or sets the default filter type (by value or by condition).

Type
`FilterType`

fields

Gets the list of **PivotField** objects exposed by the data source.

This list is created automatically whenever the **itemsSource** property is set.

Pivot views are defined by copying fields from this list to the lists that define the view: **valueFields**, **rowFields**, **columnFields**, and **filterFields**.

For example, the code below assigns a data source to the **PivotEngine** and then defines a view by adding fields to the **rowFields**, **columnFields**, and **valueFields** lists.

```
// create pivot engine
var pe = new wijmo.olap.PivotEngine();

// set data source (populates fields list)
pe.itemsSource = this.getRawData();

// prevent updates while building Olap view
pe.beginUpdate();

// show countries in rows
pe.rowFields.push('Country');

// show categories and products in columns
pe.columnFields.push('Category');
pe.columnFields.push('Product');

// show total sales in cells
pe.valueFields.push('Sales');

// done defining the view
pe.endUpdate();
```

Type
PivotFieldCollection

filterFields

Gets the list of **PivotField** objects that define the fields used as filters.

Fields on this list do not appear in the output table, but are still used for filtering the input data.

Type
PivotFieldCollection

isUpdating

Gets a value that indicates whether the engine is currently being updated.

Type
boolean

isViewDefined

Gets a value that determines whether a pivot view is currently defined.

A pivot view is defined if the **`valueFields`** list is not empty and either the **`rowFields`** or **`columnFields`** lists are not empty.

Type
boolean

itemsSource

Gets or sets the array or **`ICollectionView`** that contains the data to be analyzed, or a string containing the URL for a ComponentOne DataEngine service.

ComponentOne DataEngine services allow you to analyze large datasets on a server without downloading the raw data to the client. You can use our high-performance FlexPivot services or interface with Microsoft's SQL Server Analysis Services OLAP Cubes.

The **`PivotEngine`** sends view definitions to the server, where summaries are calculated and returned to the client.

For more information about the ComponentOne DataEngine services please refer to the **online documentation**.

Type
any

pivotView

Gets the **`ICollectionView`** containing the output pivot view.

Type
`ICollectionView`

rowFields

Gets the list of **PivotField** objects that define the fields shown as rows in the output table.

Type
PivotFieldCollection

serverMaxDetail

Gets or sets the maximum number of records the **getDetail** method should retrieve from the server.

The default value for this property is 1000, which provides a reasonable amount of detail in many scenarios. If you want to allow more detail records to be retrieved, increase the value of this property.

Type
number

serverPollInterval

Gets or sets the amount of time, in milliseconds, that the engine should wait before polling the server for progress status while retrieving results.

The default value for this property is 500, which causes the engine to poll the server for a status update every half second.

Type
number

serverTimeout

Gets or sets the maximum amount of time, in milliseconds, that the engine should wait for the results to come back from the server.

The default value for this property is 60000, equivalent to sixty seconds. If you expect server operations to take longer than that to complete, set the property to a higher value.

Type
number

showColumnTotals

Gets or sets a value that determines whether the output **pivotView** should include columns containing subtotals or grand totals.

Type
ShowTotals

showRowTotals

Gets or sets a value that determines whether the output **`pivotView`** should include rows containing subtotals or grand totals.

Type
`ShowTotals`

showZeros

Gets or sets a value that determines whether the Olap output table should use zeros to indicate the missing values.

Type
boolean

totalsBeforeData

Gets or sets a value that determines whether row and column totals should be displayed before or after regular data rows and columns.

If this value is set to true, total rows appear above data rows and total columns appear on the left of regular data columns.

Type
boolean

valueFields

Gets the list of **`PivotField`** objects that define the fields summarized in the output table.

Type
`PivotFieldCollection`

viewDefinition

Gets or sets the current pivot view definition as a JSON string.

This property is typically used to persist the current view as an application setting.

For example, the code below implements two functions that save and load view definitions using local storage:

```
// save/load views
function saveView() {
  localStorage.viewDefinition = pivotEngine.viewDefinition;
}
function loadView() {
  pivotEngine.viewDefinition = localStorage.viewDefinition;
}
```

Type
string

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends the refresh processes until next call to the **endUpdate**.

Returns
void

cancelPendingUpdates

```
cancelPendingUpdates(): void
```

Cancels any pending asynchronous view updates.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Returns
void

editField

```
editField(field: PivotField): void
```

Shows a settings dialog where users can edit a field's settings.

Parameters

- **field**: **PivotField**
  PivotField to be edited.

Returns
void

endUpdate

```
endUpdate(): void
```

Resumes refresh processes suspended by calls to **beginUpdate**.

Returns
void

getDetail

```
getDetail(item: any, binding: string): any[]
```

Gets an array containing the records summarized by a property in the **pivotView** list.

If the engine is connected to a PivotEngine server, the value returned is an **ObservableArray** that is populated asynchronously.

Parameters

- **item**: **any**
  Data item in the **pivotView** list.

- **binding**: **string**
  Name of the property being summarized.

Returns
any[]

getDetailView

```
getDetailView(item: any, binding: string): ICollectionView
```

Gets an **ICollectionView** containing the records summarized by a property in the **pivotView** list.

Parameters

- **item**: **any**
  Data item in the **pivotView** list.

- **binding**: **string**
  Name of the property being summarized.

Returns
ICollectionView

getKeys

```
getKeys(item: any, binding: string): any
```

Gets an object with information about a property in the **pivotView** list.

The object returned has two properties, 'rowKey' and 'colKey'. Each of these contains two arrays, 'fields' and 'values'. Together, this information uniquely identifies a value summarized by the **PivotEngine**.

For example, calling **getKeys** against a pivot view with two row fields 'Product' and 'Country', and a single column field 'Active' would return an object such as this one:

```
{
    rowKey: {
        fields: [ 'Product', 'Country'],
        values: [ 'Aoba', 'Japan' ]
    },
    colKey: {
        fields: [ 'Active' ],
        values: [ true ]
    }
}
```

The object identifies the subset of data used to obtain one summary value. In this case, this value represents all data items for product 'Aoba' sold in Japan with Active state set to true.

Parameters

- **item**: **any**
  Data item in the **pivotView** list.

- **binding**: **string**
  Name of the property being summarized.

Returns
any

invalidate

```
invalidate(): void
```

Invalidates the view causing an asynchronous refresh.

Returns
void

onError

```
onError(e: RequestErrorEventArgs): boolean
```

Raises the **error** event.

Parameters

- **e**: **RequestErrorEventArgs**
  RequestErrorEventArgs that contains information about the error.

Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onUpdatingView

```
onUpdatingView(e: ProgressEventArgs): void
```

Raises the **updatingView** event.

Parameters

- **e**: **ProgressEventArgs**
  ProgressEventArgs that provides the event data.

Returns
void

onViewDefinitionChanged

```
onViewDefinitionChanged(e?: EventArgs): void
```

Raises the **viewDefinitionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(force?: boolean): void
```

Summarizes the data and updates the output **pivotView**.

Parameters

- **force**: **boolean** OPTIONAL
  Refresh even while updating (see **beginUpdate**).

Returns
void

removeField

```
removeField(field: PivotField): void
```

Removes a field from the current view.

Parameters

- **field**: **PivotField**
  `PivotField` to be removed.

Returns
void

Events

error

Occurs when there is an error getting data from the server.

Arguments
`RequestErrorEventArgs`

itemsSourceChanged

Occurs after the value of the `itemsSource` property changes.

Arguments
`EventArgs`

updatedView

Occurs after the engine has finished updating the `pivotView` list.

Arguments
`EventArgs`

updatingView

Occurs when the engine starts updating the **`pivotView`** list.

Arguments
`ProgressEventArgs`

viewDefinitionChanged

Occurs after the view definition changes.

Arguments
`EventArgs`

PivotField Class

File
`wijmo.olap.js`
Module
wijmo.olap
Derived Classes
CubePivotField

Represents a property of the items in the wijmo.olap data source.

Constructor

- constructor

Properties

- aggregate
- binding
- collectionView
- dataType
- descending
- engine
- filter
- format
- header
- isActive
- isContentHtml
- key
- parentField
- showAs
- sortComparer
- weightField
- width
- wordWrap

Methods

- onPropertyChanged

Events

- propertyChanged

Constructor

constructor

```
constructor(engine: PivotEngine, binding: string, header?: string, options?: any): PivotField
```

Initializes a new instance of the **PivotField** class.

Parameters

- **engine**: **PivotEngine**
  PivotEngine that owns this field.
- **binding**: **string**
  Property that this field is bound to.
- **header**: **string** OPTIONAL
  Header shown to identify this field (defaults to the binding).
- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the field.

Returns
PivotField

Properties

aggregate

Gets or sets how the field should be summarized.

Type
Aggregate

binding

Gets or sets the name of the property the field is bound to.

Type
string

collectionView

Gets the **`ICollectionView`** bound to this field.

Type
`ICollectionView`

dataType

Gets or sets the data type of the field.

Type
`DataType`

descending

Gets or sets a value that determines whether keys should be sorted in descending order for this field.

Type
boolean

engine

Gets a reference to the **`PivotEngine`** that owns this **`PivotField`**.

Type
`PivotEngine`

filter

Gets a reference to the **`PivotFilter`** used to filter values for this field.

Type
`PivotFilter`

format

Gets or sets the format to use when displaying field values.

Type
string

header

Gets or sets a string used to represent this field in the user interface.

Type
string

isActive

Gets or sets a value that determines whether this field is currently being used in the view.

Setting this property to true causes the field to be added to the view's **`rowFields`** or **`valueFields`**, depending on the field's data type.

Type
boolean

isContentHtml

Gets or sets a value indicating whether items in this field contain HTML content rather than plain text.

Type
boolean

key

Gets the key for this **`PivotField`**.

For regular fields, the key is the field's **`header`**; for **`CubePivotField`** instances, the key is the field's **`binding`**.

Type
string

parentField

Gets this field's parent field.

When you drag the same field into the Values list multiple times, copies of the field are created so you can use the same binding with different parameters. The copies keep a reference to their parent fields.

Type
`PivotField`

showAs

Gets or sets how the field results should be formatted.

Type
`ShowAs`

sortComparer

Gets or sets a function used to compare values when sorting.

If provided, the sort comparer function should take as parameters two values of any type, and should return -1, 0, or +1 to indicate whether the first value is smaller than, equal to, or greater than the second. If the sort comparer returns null, the standard built-in comparer is used.

This **`sortComparer`** property allows you to use custom comparison algorithms that in some cases result in sorting sequences that are more consistent with user's expectations than plain string comparisons.

The example below shows a typical use for the **`sortComparer`** property:

```
// define list of products
app.products = 'Wijmo,Aoba,Olap,Xuni'.split(',');

// sort products by position in the 'app.products' array
ng.viewDefinitionChanged.addHandler(function () {
  var fld = ng.fields.getField('Product');
  if (fld) {
    fld.sortComparer = function (val1, val2) {
      return app.products.indexOf(val1) - app.products.indexOf(val2);
    }
  }
});
```

Type
Function

weightField

Gets or sets the **PivotField** used as a weight for calculating aggregates on this field.

If this property is set to null, all values are assumed to have weight one.

This property allows you to calculate weighted averages and totals. For example, if the data contains a 'Quantity' field and a 'Price' field, you could use the 'Price' field as a value field and the 'Quantity' field as a weight. The output would contain a weighted average of the data.

Type
PivotField

width

Gets or sets the preferred width to be used for showing this field in the user interface.

Type
number

wordWrap

Gets or sets a value that indicates whether the content of this field should be allowed to wrap within cells.

Type
boolean

Methods

onPropertyChanged

```
onPropertyChanged(e: PropertyChangedEventArgs): void
```

Raises the **propertyChanged** event.

Parameters

- **e**: **PropertyChangedEventArgs**
  PropertyChangedEventArgs that contains the property name, old, and new values.

Returns
void

Events

propertyChanged

Occurs when the value of a property in this **Range** changes.

Arguments
`PropertyChangedEventArgs`

PivotFieldCollection Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
ObservableArray

Represents a collection of **PivotField** objects.

Constructor

- constructor

Properties

- engine
- isUpdating
- maxItems

Methods

- beginUpdate
- clear
- deferUpdate
- endUpdate
- getField
- implementsInterface
- indexOf
- insert
- onCollectionChanged
- push
- remove
- removeAt
- setAt
- slice
- sort
- splice

Events

- collectionChanged

Constructor

constructor

```
constructor(engine: PivotEngine): PivotFieldCollection
```

Initializes a new instance of the **PivotFieldCollection** class.

Parameters

- **engine**: **PivotEngine**
  PivotEngine that owns this **PivotFieldCollection**.

Returns
PivotFieldCollection

Properties

engine

Gets a reference to the **PivotEngine** that owns this **PivotFieldCollection**.

Type
PivotEngine

isUpdating

Gets a value that indicates whether notifications are currently suspended (see **beginUpdate** and **endUpdate**).

Inherited From
ObservableArray
Type

maxItems

Gets or sets the maximum number of fields allowed in this collection.

This property is set to null by default, which means any number of items is allowed.

Type
number

Methods

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
ObservableArray
Returns
void

clear

```
clear(): void
```

Removes all items from the array.

Inherited From
ObservableArray
Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The collection will not be refreshed until the function finishes. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed without updates.

Inherited From
ObservableArray
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by a call to **beginUpdate**.

Inherited From
ObservableArray
Returns
void

getField

```
getField(key: string): PivotField
```

Gets a field by key.

Parameters

- **key**: **string**
  key to look for.

Returns
PivotField

implementsInterface

```
implementsInterface(interfaceName: string): boolean
```

Returns true if the caller queries for a supported interface.

Parameters

- **interfaceName**: **string**
  Name of the interface to look for.

Inherited From
ObservableArray
Returns
boolean

indexOf

```
indexOf(searchElement: any, fromIndex?: number): number
```

Searches for an item in the array.

Parameters

- **searchElement**: **any**
  Element to locate in the array.

- **fromIndex**: **number** OPTIONAL
  The index where the search should start.

Inherited From
ObservableArray
Returns
number

insert

```
insert(index: number, item: any): void
```

Inserts an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be added.

- **item**: **any**
  Item to add to the array.

Inherited From
ObservableArray
Returns
void

onCollectionChanged

```
onCollectionChanged(e?: NotifyCollectionChangedEventArgs): void
```

Raises the **collectionChanged** event.

Parameters

- **e**: **NotifyCollectionChangedEventArgs** OPTIONAL
  Contains a description of the change.

Inherited From
ObservableArray
Returns
void

push

```
push(...item: any[]): number
```

Overridden to allow pushing fields by header.

Parameters

- **...item**: **any**[]
  One or more **PivotField** objects to add to the array.

Returns
number

remove

```
remove(item: any): boolean
```

Removes an item from the array.

Parameters

- **item**: **any**
  Item to remove.

Inherited From
ObservableArray
Returns
boolean

removeAt

```
removeAt(index: number): void
```

Removes an item at a specific position in the array.

Parameters

- **index**: **number**
  Position of the item to remove.

Inherited From
ObservableArray
Returns
void

setAt

```
setAt(index: number, item: any): void
```

Assigns an item at a specific position in the array.

Parameters

- **index**: **number**
  Position where the item will be assigned.

- **item**: **any**
  Item to assign to the array.

Inherited From
ObservableArray
Returns
void

slice

```
slice(begin?: number, end?: number): any[]
```

Creates a shallow copy of a portion of an array.

Parameters

- **begin**: **number** OPTIONAL
  Position where the copy starts.

- **end**: **number** OPTIONAL
  Position where the copy ends.

Inherited From
ObservableArray
Returns
any[]

sort

```
sort(compareFn?: Function): this
```

Sorts the elements of the array in place.

Parameters

- **compareFn**: **Function** OPTIONAL
  Specifies a function that defines the sort order. If specified, the function should take two arguments and should return -1, +1, or 0 to indicate the first argument is smaller, greater than, or equal to the second argument. If omitted, the array is sorted in dictionary order according to the string conversion of each element.

Inherited From
ObservableArray
Returns
this

splice

```
splice(index: number, count: number, item?: any): any[]
```

Removes and/or adds items to the array.

Parameters

- **index**: **number**
  Position where items will be added or removed.

- **count**: **number**
  Number of items to remove from the array.

- **item**: **any** OPTIONAL
  Item to add to the array.

Inherited From
ObservableArray
Returns
any[]

Events

collectionChanged

Occurs when the collection changes.

Inherited From
`ObservableArray`
Arguments
`NotifyCollectionChangedEventArgs`

PivotFieldEditor Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
Control

Editor for **PivotField** objects.

Constructor

- constructor

Properties

- controlTemplate
- field
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateField

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, options?): PivotFieldEditor
```

Initializes a new instance of the **PivotFieldEditor** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
PivotFieldEditor

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **PivotFieldEditor** controls.

Type
any

field

Gets or sets a reference to the **PivotField** being edited.

Type
PivotField

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

updateEditor

```
updateEditor(): void
```

Updates editor to reflect the current field values.

Returns
void

updateField

```
updateField(): void
```

Updates field to reflect the current editor values.

Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

PivotFilter Class

File
`wijmo.olap.js`
Module
wijmo.olap

Represents a filter used to select values for a **PivotField**.

Constructor

- constructor

Properties

- conditionFilter
- filterType
- isActive
- valueFilter

Methods

- apply
- clear

Constructor

constructor

```
constructor(field: PivotField): PivotFilter
```

Initializes a new instance of the **`PivotFilter`** class.

Parameters

- **field**: **PivotField**
  `PivotField` that owns this filter.

Returns
PivotFilter

Properties

conditionFilter

Gets the **`ConditionFilter`** in this **`PivotFilter`**.

Type
`ConditionFilter`

filterType

Gets or sets the types of filtering provided by this filter.

Setting this property to null causes the filter to use the value defined by the owner filter's **`defaultFilterType`** property.

Type
`FilterType`

isActive

Gets a value that indicates whether the filter is active.

Type
boolean

valueFilter

Gets the **`ValueFilter`** in this **`PivotFilter`**.

Type
`ValueFilter`

Methods

apply

```
apply(value): boolean
```

Gets a value that indicates whether a value passes the filter.

Parameters

- **value**:
  The value to test.

Returns
boolean

clear

```
clear(): void
```

Clears the filter.

Returns
void

PivotFilterEditor Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
Control

Editor for **PivotFilter** objects.

Constructor

- constructor

Properties

- controlTemplate
- field
- filter
- hostElement
- isDisabled
- isTouching
- isUpdating
- rightToLeft

Methods

- addEventListener
- applyTemplate
- beginUpdate
- clearEditor
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onFinishEditing
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener
- updateEditor
- updateFilter

Events

- finishEditing
- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, field: PivotField, options?: any): PivotFilterEditor
```

Initializes a new instance of the **ColumnFilterEditor** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **field**: **PivotField**
  The **PivotField** to edit.
- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the editor.

Returns
PivotFilterEditor

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **PivotFilterEditor** controls.

Type
any

field

Gets a reference to the **PivotField** whose filter is being edited.

Type
PivotField

filter

Gets a reference to the **PivotFilter** being edited.

Type
PivotFilter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

clearEditor

```
clearEditor(): void
```

Clears the editor fields without applying changes to the filter.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onFinishEditing

```
onFinishEditing(e?: CancelEventArgs): void
```

Raises the **finishEditing** event.

Parameters

- **e**: **CancelEventArgs** OPTIONAL

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

updateEditor

```
updateEditor(): void
```

Updates the editor with current filter settings.

Returns
void

updateFilter

```
updateFilter(): void
```

Updates the filter to reflect the current editor values.

Returns
void

Events

finishEditing

Occurs when the user finishes editing the filter.

Arguments
`CancelEventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

PivotGrid Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
FlexGrid
Derived Classes
WjPivotGrid

Extends the **FlexGrid** control to display pivot tables.

To use this control, set its **itemsSource** property to an instance of a **PivotPanel** control or to a **PivotEngine**.

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoSizeMode
- bottomLeftCells
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsibleSubtotals
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- customContextMenu
- deferResizing
- editableCollectionView
- editRange
- engine
- frozenColumns
- frozenRows
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- isDisabled
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemValidator
- keyActionEnter
- keyActionTab
- mergeManager
- newRowAtTop
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- rightToLeft
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionMode
- showAlternatingRows
- showColumnFieldHeaders
- showDetailOnDoubleClick
- showDropDown
- showErrors
- showGroups
- showMarquee
- showRowFieldHeaders
- showRowFieldSort
- showSelectedHeaders
- showSort
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- validateEdits
- viewRange
- virtualizationThreshold

Methods

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows

- beginUpdate
- canEditCell
- collapseColumnsToLevel
- collapseGroupsToLevel
- collapseRowsToLevel
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getDetail
- getDetailView
- getKeys
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- showDetail
- startEditing
- toggleDropDownList

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn

- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): PivotGrid
```

Initializes a new instance of the **PivotGrid** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
PivotGrid

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Inherited From
`FlexGrid`
Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Inherited From
`FlexGrid`
Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Inherited From
`FlexGrid`
Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Inherited From
`FlexGrid`
Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Inherited From
`FlexGrid`
Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Inherited From
`FlexGrid`
Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **`itemsSource`**.

The column generation depends on the **`itemsSource`** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **`autoGenerateColumns`** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Inherited From
`FlexGrid`
Type
boolean

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Inherited From
`FlexGrid`
Type
`AutoSizeMode`

bottomLeftCells

Gets the **`GridPanel`** that contains the bottom left cells.

The **`bottomLeftCells`** panel appears below the row headers, to the left of the **`columnFooters`** panel.

Inherited From
`FlexGrid`
Type
`GridPanel`

cellFactory

Gets or sets the **`CellFactory`** that creates and updates cells for this grid.

Inherited From
`FlexGrid`
Type
`CellFactory`

cells

Gets the **`GridPanel`** that contains the data cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

centerHeadersVertically

Gets or sets a value that determines whether the content of header cells should be vertically centered.

Type
boolean

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Inherited From
`FlexGrid`
Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Inherited From
`FlexGrid`
Type
`Size`

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Inherited From
`FlexGrid`
Type
boolean

collapsibleSubtotals

Gets or sets a value that determines whether the grid should allow users to collapse and expand subtotal groups of rows and columns.

Type
boolean

collectionView

Gets the **`ICollectionView`** that contains the grid data.

Inherited From
`FlexGrid`
Type
`ICollectionView`

columnFooters

Gets the **GridPanel** that contains the column footer cells.

The **columnFooters** panel appears below the grid cells, to the right of the **bottomLeftCells** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **columnFooters** panel to display summary data for columns that have the **aggregate** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Inherited From
FlexGrid
Type
GridPanel

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Inherited From
FlexGrid
Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Inherited From
`FlexGrid`
Type
string

columns

Gets the grid's column collection.

Inherited From
`FlexGrid`
Type
`ColumnCollection`

controlRect

Gets the bounding rectangle of the control in page coordinates.

Inherited From
`FlexGrid`
Type
`Rect`

STATIC controlTemplate

Gets or sets the template used to instantiate `FlexGrid` controls.

Inherited From
`FlexGrid`
Type
any

customContextMenu

Gets or sets a value that determines whether the grid should provide a custom context menu.

The custom context menu includes commands for changing field settings, removing fields, or showing detail records for the grid cells.

Type
boolean

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **`deferResizing`** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Inherited From
`FlexGrid`
Type
boolean

editableCollectionView

Gets the **`IEditableCollectionView`** that contains the grid data.

Inherited From
`FlexGrid`
Type
`IEditableCollectionView`

editRange

Gets a **`CellRange`** that identifies the cell currently being edited.

Inherited From
`FlexGrid`
Type
`CellRange`

engine

Gets a reference to the **PivotEngine** that owns this **PivotGrid**.

Type
PivotEngine

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Inherited From
FlexGrid
Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Inherited From
FlexGrid
Type
number

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Inherited From
`FlexGrid`
Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Inherited From
`FlexGrid`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Inherited From
`FlexGrid`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Inherited From
FlexGrid
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** that contains items shown on the grid.

Inherited From
FlexGrid
Type
any

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **getError** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Inherited From
FlexGrid
Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **MoveDown**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Inherited From
`FlexGrid`
Type
`KeyAction`

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **None**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **Cycle**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **CycleOut** setting that causes the selection to move through the cells (as **Cycle**), and then on to the next/previous control on the page when the last or first cells are selected.

Inherited From
`FlexGrid`
Type
`KeyAction`

mergeManager

Gets or sets the **MergeManager** object responsible for determining how cells should be merged.

Inherited From
`FlexGrid`
Type
`MergeManager`

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **`newRowAtTop`** property to true, and you want the new row template to remain visible at all times, set the **`frozenRows`** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **`allowAddNew`** property is set to true and if the **`itemsSource`** object supports adding new items.

Inherited From
`FlexGrid`
Type
boolean

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **`preserveOutlineState`** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Inherited From
`FlexGrid`
Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Inherited From
`FlexGrid`
Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Inherited From
`FlexGrid`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Inherited From
`FlexGrid`
Type
string

rowHeaders

Gets the **`GridPanel`** that contains the row header cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

rows

Gets the grid's row collection.

Inherited From
`FlexGrid`
Type
`RowCollection`

scrollPosition

Gets or sets a **`Point`** that represents the value of the grid's scrollbars.

Inherited From
`FlexGrid`
Type
`Point`

scrollSize

Gets the size of the grid content in pixels.

Inherited From
`FlexGrid`
Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when **`selectionMode`** is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when **`selectionMode`** is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selection

Gets or sets the current selection.

Inherited From
`FlexGrid`
Type
`CellRange`

selectionMode

Gets or sets the current selection mode.

Inherited From
`FlexGrid`
Type
`SelectionMode`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Inherited From
`FlexGrid`
Type
boolean

showColumnFieldHeaders

Gets or sets a value that determines whether the grid should display column field headers in its top-left panel.

Type
boolean

showDetailOnDoubleClick

Gets or sets a value that determines whether the grid should show a popup containing the detail records when the user double-clicks a cell.

Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **showDropDown** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
FlexGrid
Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **itemValidator** property or the **getError** property on the grid's **itemsSource**.

Inherited From
FlexGrid
Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Inherited From
`FlexGrid`
Type
boolean

showRowFieldHeaders

Gets or sets a value that determines whether the grid should display row field headers in its top-left panel.

Type
boolean

showRowFieldSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers for row fields.

Unlike regular column headers, row fields are always sorted, either in ascending or descending order. If you set this property to true, sort icons will always be displayed over any row field headers.

Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Inherited From
`FlexGrid`
Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Inherited From
`FlexGrid`
Type
boolean

topLeftCells

Gets the **GridPanel** that contains the top left cells (to the left of the column headers).

Inherited From
FlexGrid
Type
GridPanel

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Inherited From
FlexGrid
Type
number

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **getError** method on the grid's **itemsSource**.

Inherited From
FlexGrid
Type
boolean

viewRange

Gets the range of cells currently in view.

Inherited From
FlexGrid
Type
CellRange

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Inherited From
`FlexGrid`
Type
number

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.

- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`Control`
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
FlexGrid
Returns
void

collapseColumnsToLevel

```
collapseColumnsToLevel(level: number): void
```

Collapses all columns to a given level.

Parameters

- **level**: **number**
  Maximum column level to show. Zero means show only grand totals; one means show only top-level groups; very high levels expand all columns.

Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Inherited From
FlexGrid
Returns
void

collapseRowsToLevel

```
collapseRowsToLevel(level: number): void
```

Collapses all rows to a given level.

Parameters

- **level**: **number**
  Maximum row level to show. Zero means show only grand totals; one means show only top-level groups; very high levels expand all rows.

Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Inherited From
`FlexGrid`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Inherited From
FlexGrid
Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Inherited From
`FlexGrid`
Returns
void

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
`FlexGrid`
Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Inherited From
FlexGrid
Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Inherited From
`FlexGrid`
Returns
`Column`

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDetail

```
getDetail(row: number, col: number): any[]
```

Gets an array containing the records summarized by a given grid cell.

Parameters

- **row**: **number**
  Index of the row that contains the cell.
- **col**: **number**
  Index of the column that contains the cell.

Returns
any[]

getDetailView

```
getDetailView(row: number, col: number): ICollectionView
```

Gets an **`ICollectionView`** containing the records summarized by a given grid cell.

Parameters

- **row**: **number**
  Index of the row that contains the cell.
- **col**: **number**
  Index of the column that contains the cell.

Returns
`ICollectionView`

getKeys

```
getKeys(row: number, col: number): any
```

Gets an object with information about the fields and values being used to summarize a given cell.

For more details, see the @PivotEngine.getKeys method.

Parameters

- **row**: **number**
  Index of the row that contains the cell.
- **col**: **number**
  Index of the column that contains the cell.

Returns
any

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Inherited From
FlexGrid
Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **`SelectedState`** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.

- **c**: **number**
  Column index of the cell to inspect.

Inherited From
`FlexGrid`
Returns
`SelectedState`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Inherited From
FlexGrid
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Inherited From
FlexGrid
Returns
boolean

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the grid layout and content, or just the content.

Inherited From
FlexGrid
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.

- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.

- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Inherited From
FlexGrid
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.
- **c**: **number**
  Index of the column to scroll into view.

Inherited From
FlexGrid
Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

Parameters

- **rng**: **any**
  Range to select.
- **show**: **any** OPTIONAL
  Whether to scroll the new selection into view.

Inherited From
FlexGrid
Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Sets the value of a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the grid to show the change.

Inherited From
`FlexGrid`
Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Hidden rows and columns are skipped.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
void

showDetail

```
showDetail(row: number, col: number): void
```

Shows a dialog containing details for a given grid cell.

Parameters

- **row**: **number**
  Index of the row that contains the cell.

- **col**: **number**
  Index of the column that contains the cell.

Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Inherited From
FlexGrid
Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Inherited From
`FlexGrid`
Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Inherited From
`FlexGrid`
Arguments
`CellEditEndingEventArgs`

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **allowDelete** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **allowDelete** property).

The event handler may cancel the row deletion.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedColumn

Occurs when the user finishes dragging a column.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedRow

Occurs when the user finishes dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggingRowOver

Occurs as the user drags a row to a new position.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **itemFormatter** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Inherited From
FlexGrid
Arguments
FormatItemEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

resizedColumn

Occurs when the user finishes resizing a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizedRow

Occurs when the user finishes resizing rows.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **`rowEditStarted`** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarted

Occurs after a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

selectionChanged

Occurs after selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

updatedLayout

Occurs after the grid has updated its internal layout.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatingLayout

Occurs before the grid updates its internal layout.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

PivotPanel Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
Control
Derived Classes
WjPivotPanel

Provides a user interface for interactively transforming regular data tables into Olap pivot tables.

Olap pivot tables group data into one or more dimensions. The dimensions are represented by rows and columns on a grid, and the summarized data is stored in the grid cells.

Use the **itemsSource** property to set the source data, and the **pivotView** property to get the output table containing the summarized data.

Constructor

- constructor

Properties

- autoGenerateFields
- collectionView
- columnFields
- controlTemplate
- engine
- fields
- filterFields
- hostElement
- isDisabled
- isTouching
- isUpdating
- isViewDefined
- itemsSource
- pivotView
- rightToLeft
- rowFields
- valueFields
- viewDefinition

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onItemsSourceChanged
- onLostFocus
- onUpdatedView
- onUpdatingView
- onViewDefinitionChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- itemsSourceChanged
- lostFocus
- updatedView
- updatingView
- viewDefinitionChanged

Constructor

constructor

```
constructor(element: any, options?): PivotPanel
```

Initializes a new instance of the **PivotPanel** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Returns
PivotPanel

Properties

autoGenerateFields

Gets or sets a value that determines whether the engine should populate the **fields** collection automatically based on the **itemsSource**.

Type
boolean

collectionView

Gets the **ICollectionView** that contains the raw data.

Type
ICollectionView

columnFields

Gets the list of fields that define the columns in the output table.

Type
PivotFieldCollection

STATIC controlTemplate

Gets or sets the template used to instantiate **PivotPanel** controls.

Type
any

engine

Gets or sets the **PivotEngine** being controlled by this **PivotPanel**.

Type
PivotEngine

fields

Gets the list of fields available for building views.

Type
PivotFieldCollection

filterFields

Gets the list of fields that define filters applied while generating the output table.

Type
PivotFieldCollection

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

isViewDefined

Gets a value that determines whether a pivot view is currently defined.

A pivot view is defined if the **`valueFields`** list is not empty and either the **`rowFields`** or **`columnFields`** lists are not empty.

Type
boolean

itemsSource

Gets or sets the array or **`ICollectionView`** that contains the raw data.

Type
any

pivotView

Gets the **`ICollectionView`** containing the output pivot view.

Type
`ICollectionView`

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowFields

Gets the list of fields that define the rows in the output table.

Type
`PivotFieldCollection`

valueFields

Gets the list of fields that define the values shown in the output table.

Type
`PivotFieldCollection`

viewDefinition

Gets or sets the current pivot view definition as a JSON string.

This property is typically used to persist the current view as an application setting.

For example, the code below implements two functions that save and load view definitions using local storage:

```
// save/load views
function saveView() {
  localStorage.viewDefinition = pivotPanel.viewDefinition;
}
function loadView() {
  pivotPanel.viewDefinition = localStorage.viewDefinition;
}
```

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

onUpdatingView

```
onUpdatingView(e: ProgressEventArgs): void
```

Raises the **updatingView** event.

Parameters

- **e**: **ProgressEventArgs**
  ProgressEventArgs that provides the event data.

Returns
void

onViewDefinitionChanged

```
onViewDefinitionChanged(e?: EventArgs): void
```

Raises the **viewDefinitionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

itemsSourceChanged

Occurs after the value of the **`itemsSource`** property changes.

Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

updatedView

Occurs after the engine has finished updating the **`pivotView`** list.

Arguments
`EventArgs`

updatingView

Occurs when the engine starts updating the **`pivotView`** list.

Arguments
`ProgressEventArgs`

viewDefinitionChanged

Occurs after the view definition changes.

Arguments
`EventArgs`

ProgressEventArgs Class

File
`wijmo.olap.js`
Module
wijmo.olap
Base Class
EventArgs

Provides arguments for progress events.

Constructor

- constructor

Properties

- empty
- progress

Constructor

constructor

```
constructor(progress: number): ProgressEventArgs
```

Initializes a new instance of the **ProgressEventArgs** class.

Parameters

- **progress**: **number**
  Number between 0 and 100 that represents the progress.

Returns
ProgressEventArgs

Properties

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
`EventArgs`
Type
`EventArgs`

progress

Gets the current progress as a number between 0 and 100.

Type
number

DimensionType Enum

File
`wijmo.olap.js`
Module
wijmo.olap

Defines the dimension type of a **CubePivotField**.

Members

Name	Value	Description
Dimension	0	Fields that contain categories used to summarize data.
Measure	1	Fields that contain quantitative, numerical information.
Kpi	2	Calculations associated with a measure group used to evaluate business success.
NameSet	3	Multidimensional Expression (MDX) that returns a set of dimension members.
Attribute	4	Provide supplementary information about dimension members.
Folder	5	Used to categorize measures and improve the user browsing experience.
Hierarchy	6	Metadata that define relationships between two or more columns in a table.
Date	7	Dimension with time-based levels of granularity for analysis and reporting.
Currency	8	Dimension whose attributes represent a list of currencies for financial reporting purposes.

LegendVisibility Enum

File
`wijmo.olap.js`
Module
wijmo.olap

Specifies constants that define when the chart legend should be displayed.

Members

Name	Value	Description
Always	0	Always show the legend.
Never	1	Never show the legend.
Auto	2	Show the legend if the chart has more than one series.

PivotChartType Enum

File
`wijmo.olap.js`
Module
wijmo.olap

Specifies constants that define the chart type.

Members

Name	Value	Description
Column	0	Shows vertical bars and allows you to compare values of items across categories.
Bar	1	Shows horizontal bars.
Scatter	2	Shows patterns within the data using X and Y coordinates.
Line	3	Shows trends over a period of time or across categories.
Area	4	Shows line chart with the area below the line filled with color.
Pie	5	Shows pie chart.

ShowAs Enum

File
`wijmo.olap.js`
Module
wijmo.olap

Specifies constants that define calculations to be applied to cells in the output view.

Members

Name	Value	Description
NoCalculation	0	Show plain aggregated values.
DiffRow	1	Show differences between each item and the item in the previous row.
DiffRowPct	2	Show differences between each item and the item in the previous row as a percentage.
DiffCol	3	Show differences between each item and the item in the previous column.
DiffColPct	4	Show differences between each item and the item in the previous column as a percentage.
PctGrand	5	Show values as a percentage of the grand totals for the field.
PctRow	6	Show values as a percentage of the row totals for the field.
PctCol	7	Show values as a percentage of the column totals for the field.
RunTot	8	Show values as running totals.
RunTotPct	9	Show values as percentage running totals.

ShowTotals Enum

File
`wijmo.olap.js`
Module
wijmo.olap

Specifies constants that define whether to include totals in the output table.

Members

Name	Value	Description
None	0	Do not show any totals.
GrandTotals	1	Show grand totals.
Subtotals	2	Show subtotals and grand totals.

wijmo.viewer Module

File
`wijmo.viewer.js`
Module
wijmo.viewer

Defines a series of classes, interfaces and functions related to the viewer controls.

Classes

- PdfViewer
- QueryLoadingDataEventArgs
- ReportViewer
- ViewerBase

Interfaces

- ICatalogItem
- IPromise

Enums

- CatalogItemType
- MouseMode
- ViewMode
- ZoomMode

PdfViewer Class

File
wijmo.viewer.js
Module
wijmo.viewer
Base Class
ViewerBase
Derived Classes
WjPdfViewer

Defines the PDFViewer control for displaying the PDF document.

The **serviceUrl** property indicates the url of C1 Web API which provides PDF services. The PDF services use C1PdfDocumentSource to process PDF document.

Here is the sample to show a PDF document:

```
var pdfViewer = new wijmo.viewer.PdfViewer('#pdfViewer');
pdfViewer.serviceUrl= 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report';
pdfViewer.filePath= 'PdfRoot/DefaultDocument.pdf';
```

Constructor

- constructor

Properties

- controlTemplate
- filePath
- fullScreen
- hostElement
- isDisabled
- isTouching
- isUpdating
- mouseMode
- pageIndex
- rightToLeft
- selectMouseMode
- serviceUrl
- thresholdWidth
- viewMode
- zoomFactor
- zoomMode

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

Events

- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

Constructor

constructor

```
constructor(element: any, options?: any): PdfViewer
```

Initializes a new instance of the **PdfViewer** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
PdfViewer

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate the viewer controls.

Inherited From
ViewerBase
Type
any

filePath

Gets or sets the full path to the document on the server.

The path starts with the key of a provider which is registered at server for locating specified document.

Inherited From
ViewerBase
Type
string

fullScreen

Gets or sets a value indicating whether the viewer is under full screen mode.

Inherited From
`ViewerBase`
Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

mouseMode

Gets or sets a value indicating the mouse behavior.

The default is SelectTool which means clicking and dragging the mouse will select the text.

Inherited From
`ViewerBase`
Type
`MouseMode`

pageIndex

Gets the index of the page which is currently displayed in the view panel.

Inherited From
`ViewerBase`
Type
number

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectMouseMode

Deprecated: use mouseMode instead.

Inherited From
`ViewerBase`
Type
boolean

serviceUrl

Gets or sets the address of C1 Web API service.

For example, "http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report".

Inherited From
`ViewerBase`
Type
string

thresholdWidth

Gets or sets the threshold to switch between mobile and PC template.

Default value is 767px. If width of control is smaller than thresholdWidth, mobile template will be applied. If width of control is equal or greater than thresholdWidth, PC template will be applied. If thresholdWidth is set to 0, then only PC template is applied and if it's set to a large number e.g. 9999, then only mobile template is applied.

Inherited From
`ViewerBase`
Type
number

viewMode

Gets or sets a value indicating how to show the document pages.

Inherited From
`ViewerBase`
Type
`ViewMode`

zoomFactor

Gets or sets a value indicating the current zoom factor to show the document pages.

Inherited From
`ViewerBase`
Type
number

zoomMode

Gets or sets a value indicating the current zoom mode to show the document pages.

Inherited From
`ViewerBase`
Type
`ZoomMode`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

moveToPage

```
moveToPage(index: number): IPromise
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index (0-base) of the page to move to.

Inherited From
ViewerBase
Returns
IPromise

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

Raises the **fullScreenChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

Raises the **mouseModeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

Raises the **pageIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

Raises the **queryLoadingData** event.

Parameters

- **e**: **QueryLoadingDataEventArgs**
  The **QueryLoadingDataEventArgs** object that contains the loading data.

Inherited From
ViewerBase
Returns
void

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

Deprecated: use onMouseModeChanged instead.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

Raises the **viewModeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

Raises the **zoomFactorChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
`ViewerBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

reload

```
reload(): void
```

Reloads the document.

This is useful for force reloading and rerendering the document.

Inherited From
`ViewerBase`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

showPageSetupDialog

```
showPageSetupDialog(): void
```

Shows the page setup dialog.

Inherited From
`ViewerBase`
Returns
void

zoomToView

```
zoomToView(): void
```

Scales the current page to show the whole page in view panel.

Inherited From
`ViewerBase`
Returns
void

zoomToViewWidth

```
zoomToViewWidth(): void
```

Scales the current page to fit the width of the view panel.

Inherited From
`ViewerBase`
Returns
void

Events

fullScreenChanged

Occurs after the full screen mode is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

mouseModeChanged

Occurs after the mouse mode is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

pageIndexChanged

Occurs after the page index is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

queryLoadingData

Occurs when querying the request data sent to the service before loading the document.

Inherited From
`ViewerBase`
Arguments
`QueryLoadingDataEventArgs`

selectMouseModeChanged

Deprecated: use mouseModeChanged instead.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

viewModeChanged

Occurs after the view mode is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

zoomFactorChanged

Occurs after the zoom factor is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

QueryLoadingDataEventArgs Class

File
wijmo.viewer.js
Module
wijmo.viewer
Base Class
EventArgs

Provides arguments for **queryLoadingData** event.

Constructor

- constructor

Properties

- data
- empty

Constructor

constructor

```
constructor(data?: any): QueryLoadingDataEventArgs
```

Initializes a new instance of the **QueryLoadingDataEventArgs** class.

Parameters

- **data**: **any** OPTIONAL
  The request data sent to the service on loading the document.

Returns
QueryLoadingDataEventArgs

Properties

data

Gets the request data sent to the service on loading the document.

Type
any

STATIC empty

Provides a value to use with events that do not have event data.

Inherited From
EventArgs
Type
EventArgs

ReportViewer Class

File
`wijmo.viewer.js`
Module
wijmo.viewer
Base Class
ViewerBase
Derived Classes
WjReportViewer

Defines the ReportViewer control for displaying the FlexReport or SSRS report.

The **serviceUrl** property indicates the url of C1 Web API which provides report services. The report services use C1FlexReport to process a FlexReport, and use C1SSRSDocumentSource and C1PdfDocumentSource to process an SSRS report.

Here is a sample of how to show a FlexReport:

```
var reportViewer = new wijmo.viewer.ReportViewer('#reportViewer');
reportViewer.serviceUrl = 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report';
reportViewer.filePath = 'ReportsRoot/Formatting/AlternateBackground.flxr';
reportViewer.reportName = 'AlternateBackground';
```

Here is a sample of how to show an SSRS report:

```
var reportViewer = new wijmo.viewer.ReportViewer('#reportViewer');
reportViewer.serviceUrl = 'http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report';
reportViewer.filePath = 'c1ssrs/AdventureWorks/Company Sales';
```

Constructor

- constructor

Properties

- controlTemplate
- filePath
- fullScreen
- hostElement
- isDisabled
- isTouching
- isUpdating
- mouseMode
- pageIndex
- paginated
- reportName
- rightToLeft
- selectMouseMode
- serviceUrl
- thresholdWidth
- viewMode
- zoomFactor
- zoomMode

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getReportNames
- getReports
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

Events

- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

Constructor

constructor

```
constructor(element: any, options?: any): ReportViewer
```

Initializes a new instance of the **ReportViewer** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
ReportViewer

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate the viewer controls.

Inherited From
ViewerBase
Type
any

filePath

Gets or sets the full path to the document on the server.

The path starts with the key of a provider which is registered at server for locating specified document.

Inherited From
ViewerBase
Type
string

fullScreen

Gets or sets a value indicating whether the viewer is under full screen mode.

Inherited From
`ViewerBase`
Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

mouseMode

Gets or sets a value indicating the mouse behavior.

The default is SelectTool which means clicking and dragging the mouse will select the text.

Inherited From
`ViewerBase`
Type
`MouseMode`

pageIndex

Gets the index of the page which is currently displayed in the view panel.

Inherited From
`ViewerBase`
Type
number

paginated

Gets or sets a value indicating whether the content should be represented as a set of fixed sized pages.

The default value is null which means using paginated mode for a FlexReport and non-paginaged mode for an SSRS report.

Type
boolean

reportName

Gets or sets the report name.

For FlexReport, sets it with the report name defined in the FlexReport definition file. For SSRS report, leave it as empty string. The SSRS report path is specified by the **`filePath`** property.

Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectMouseMode

Deprecated: use mouseMode instead.

Inherited From
`ViewerBase`
Type
boolean

serviceUrl

Gets or sets the address of C1 Web API service.

For example, "http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report".

Inherited From
`ViewerBase`
Type
string

thresholdWidth

Gets or sets the threshold to switch between mobile and PC template.

Default value is 767px. If width of control is smaller than thresholdWidth, mobile template will be applied. If width of control is equal or greater than thresholdWidth, PC template will be applied. If thresholdWidth is set to 0, then only PC template is applied and if it's set to a large number e.g. 9999, then only mobile template is applied.

Inherited From
`ViewerBase`
Type
number

viewMode

Gets or sets a value indicating how to show the document pages.

Inherited From
`ViewerBase`
Type
`ViewMode`

zoomFactor

Gets or sets a value indicating the current zoom factor to show the document pages.

Inherited From
`ViewerBase`
Type
number

zoomMode

Gets or sets a value indicating the current zoom mode to show the document pages.

Inherited From
`ViewerBase`
Type
`ZoomMode`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

STATIC getReportNames

```
getReportNames(serviceUrl: string, reportFilePath: string): IPromise
```

Gets the report names defined in the specified FlexReport definition file.

Parameters

- **serviceUrl**: **string**
  The address of C1 Web API service.
- **reportFilePath**: **string**
  The full path to the FlexReport definition file.

Returns
IPromise

STATIC getReports

```
getReports(serviceUrl: string, path: string, data?: any): IPromise
```

Gets the catalog items in the specified folder path.

You can get all items under the folder path by passing the data parameter as: 1) A true value. 2) An object which has the "recursive" property with true value.

Parameters

- **serviceUrl**: **string**
  The address of C1 Web API service.
- **path**: **string**
  The folder path. The path to the FlexReport definition file will be treated as a folder path.
- **data**: **any** OPTIONAL
  The request data sent to the report service, or a boolean value indicates whether getting all items under the path.

Returns
IPromise

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

moveToPage

```
moveToPage(index: number): IPromise
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index (0-base) of the page to move to.

Inherited From
ViewerBase
Returns
IPromise

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

Raises the **fullScreenChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

Raises the **mouseModeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

Raises the **pageIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

Raises the **queryLoadingData** event.

Parameters

- **e**: **QueryLoadingDataEventArgs**
  The `QueryLoadingDataEventArgs` object that contains the loading data.

Inherited From
`ViewerBase`
Returns
void

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

Deprecated: use onMouseModeChanged instead.

Parameters

- **e**: **EventArgs** OPTIONAL
  The `EventArgs` object.

Inherited From
`ViewerBase`
Returns
void

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

Raises the **viewModeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

Raises the **zoomFactorChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Inherited From
ViewerBase
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
`ViewerBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

reload

```
reload(): void
```

Reloads the document.

This is useful for force reloading and rerendering the document.

Inherited From
`ViewerBase`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

showPageSetupDialog

```
showPageSetupDialog(): void
```

Shows the page setup dialog.

Inherited From
`ViewerBase`
Returns
void

zoomToView

```
zoomToView(): void
```

Scales the current page to show the whole page in view panel.

Inherited From
`ViewerBase`
Returns
void

zoomToViewWidth

```
zoomToViewWidth(): void
```

Scales the current page to fit the width of the view panel.

Inherited From
`ViewerBase`
Returns
void

Events

fullScreenChanged

Occurs after the full screen mode is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

mouseModeChanged

Occurs after the mouse mode is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

pageIndexChanged

Occurs after the page index is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

queryLoadingData

Occurs when querying the request data sent to the service before loading the document.

Inherited From
`ViewerBase`
Arguments
`QueryLoadingDataEventArgs`

selectMouseModeChanged

Deprecated: use mouseModeChanged instead.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

viewModeChanged

Occurs after the view mode is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

zoomFactorChanged

Occurs after the zoom factor is changed.

Inherited From
`ViewerBase`
Arguments
`EventArgs`

ViewerBase Class

File
`wijmo.viewer.js`
Module
wijmo.viewer
Base Class
Control
Derived Classes
PdfViewer, **ReportViewer**

Base class for all the viewer controls.

Constructor

- constructor

Properties

- controlTemplate
- filePath
- fullScreen
- hostElement
- isDisabled
- isTouching
- isUpdating
- mouseMode
- pageIndex
- rightToLeft
- selectMouseMode
- serviceUrl
- thresholdWidth
- viewMode
- zoomFactor
- zoomMode

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- moveToPage
- onFullScreenChanged
- onGotFocus
- onLostFocus
- onMouseModeChanged
- onPageIndexChanged
- onQueryLoadingData
- onSelectMouseModeChanged
- onViewModeChanged
- onZoomFactorChanged
- refresh
- refreshAll
- reload
- removeEventListener
- showPageSetupDialog
- zoomToView
- zoomToViewWidth

Events

- fullScreenChanged
- gotFocus
- lostFocus
- mouseModeChanged
- pageIndexChanged
- queryLoadingData
- selectMouseModeChanged
- viewModeChanged
- zoomFactorChanged

Constructor

constructor

```
constructor(element: any, options?: any): ViewerBase
```

Initializes a new instance of the **ViewerBase** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the control.

Returns
ViewerBase

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate the viewer controls.

Type
any

filePath

Gets or sets the full path to the document on the server.

The path starts with the key of a provider which is registered at server for locating specified document.

Type
string

fullScreen

Gets or sets a value indicating whether the viewer is under full screen mode.

Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

mouseMode

Gets or sets a value indicating the mouse behavior.

The default is SelectTool which means clicking and dragging the mouse will select the text.

Type
`MouseMode`

pageIndex

Gets the index of the page which is currently displayed in the view panel.

Type
number

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectMouseMode

Deprecated: use mouseMode instead.

Type
boolean

serviceUrl

Gets or sets the address of C1 Web API service.

For example, "http://demos.componentone.com/ASPNET/c1webapi/4.0.20172.105/api/report".

Type
string

thresholdWidth

Gets or sets the threshold to switch between mobile and PC template.

Default value is 767px. If width of control is smaller than thresholdWidth, mobile template will be applied. If width of control is equal or greater than thresholdWidth, PC template will be applied. If thresholdWidth is set to 0, then only PC template is applied and if it's set to a large number e.g. 9999, then only mobile template is applied.

Type
number

viewMode

Gets or sets a value indicating how to show the document pages.

Type
`ViewMode`

zoomFactor

Gets or sets a value indicating the current zoom factor to show the document pages.

Type
number

zoomMode

Gets or sets a value indicating the current zoom mode to show the document pages.

Type
`ZoomMode`

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate**/**endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

moveToPage

```
moveToPage(index: number): IPromise
```

Moves to the page at the specified index.

Parameters

- **index**: **number**
  Index (0-base) of the page to move to.

Returns
IPromise

onFullScreenChanged

```
onFullScreenChanged(e?: EventArgs): void
```

Raises the **fullScreenChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onMouseModeChanged

```
onMouseModeChanged(e?: EventArgs): void
```

Raises the **mouseModeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Returns
void

onPageIndexChanged

```
onPageIndexChanged(e?: EventArgs): void
```

Raises the **pageIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Returns
void

onQueryLoadingData

```
onQueryLoadingData(e: QueryLoadingDataEventArgs): void
```

Raises the **queryLoadingData** event.

Parameters

- **e**: **QueryLoadingDataEventArgs**
  The **QueryLoadingDataEventArgs** object that contains the loading data.

Returns
void

onSelectMouseModeChanged

```
onSelectMouseModeChanged(e?: EventArgs): void
```

Deprecated: use onMouseModeChanged instead.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Returns
void

onViewModeChanged

```
onViewModeChanged(e?: EventArgs): void
```

Raises the **viewModeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Returns
void

onZoomFactorChanged

```
onZoomFactorChanged(e?: EventArgs): void
```

Raises the **zoomFactorChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL
  The **EventArgs** object.

Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

reload

```
reload(): void
```

Reloads the document.

This is useful for force reloading and rerendering the document.

Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

showPageSetupDialog

```
showPageSetupDialog(): void
```

Shows the page setup dialog.

Returns
void

zoomToView

```
zoomToView(): void
```

Scales the current page to show the whole page in view panel.

Returns
void

zoomToViewWidth

```
zoomToViewWidth(): void
```

Scales the current page to fit the width of the view panel.

Returns
void

Events

fullScreenChanged

Occurs after the full screen mode is changed.

Arguments
`EventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

mouseModeChanged

Occurs after the mouse mode is changed.

Arguments
`EventArgs`

pageIndexChanged

Occurs after the page index is changed.

Arguments
`EventArgs`

queryLoadingData

Occurs when querying the request data sent to the service before loading the document.

Arguments
`QueryLoadingDataEventArgs`

selectMouseModeChanged

Deprecated: use mouseModeChanged instead.

Arguments
`EventArgs`

viewModeChanged

Occurs after the view mode is changed.

Arguments
`EventArgs`

zoomFactorChanged

Occurs after the zoom factor is changed.

Arguments
`EventArgs`

ICatalogItem Interface

File
`wijmo.viewer.js`
Module
wijmo.viewer

Describes an item in the report server of a specific path.

Properties

- items
- name
- path
- type

Properties

items

The array of child items.

Type
`ICatalogItem`[]

name

The short name of the item.

Type
string

path

The full path (starts with the report provider key) of the item.

Type
string

type

The type of the item.

Type
`CatalogItemType`

IPromise Interface

File
`wijmo.viewer.js`
Module
wijmo.viewer

Defines the interface of promise which is used for asynchronous calling.

Methods

- catch
- then

Methods

catch

```
catch(onRejected?: (reason?: any)): IPromise
```

Call the function after a promise is rejected.

Parameters

- **onRejected**: **(reason?: any)** OPTIONAL
  The function which will be executed when a promise is rejected. This has a single parameter, the rejection reason. The return value will be passed to the next callback function.

Returns
`IPromise`

then

```
then(onFulfilled?: (value?: any), onRejected?: (reason?: any)): IPromise
```

Call the function after a promise is fulfilled or rejected.

Parameters

- **onFulfilled**: **(value?: any)** OPTIONAL
  The function which will be executed when a promise is fulfilled. This has a single parameter, the fulfillment value. If a value is returned, it will be passed to the next callback function. If no value is returned, the original value will be passed.

- **onRejected**: **(reason?: any)** OPTIONAL
  The function which will be executed when a promise is rejected. This has a single parameter, the rejection reason. If a value is returned, it will be passed to the next callback function. If no value is returned, the original value will be passed.

Returns
`IPromise`

CatalogItemType Enum

File
`wijmo.viewer.js`
Module
wijmo.viewer

Specifies the type of a catalog item.

Members

Name	Value	Description
Folder	0	A folder.
File	1	A FlexReport definition file.
Report	2	An SSRS report or a FlexReport defined in the FlexReport definition file.

MouseMode Enum

File
`wijmo.viewer.js`
Module
wijmo.viewer

Specifies the mouse modes, which defines the mouse behavior of viewer.

Members

Name	Value	Description
SelectTool	0	Select text.
MoveTool	1	Move page.
RubberbandTool	2	Rubberband to zoom.
MagnifierTool	3	Magnifier tool.

ViewMode Enum

File
`wijmo.viewer.js`
Module
wijmo.viewer

Specifies the view modes, which define how to show document pages in the view panel.

Members

Name	Value	Description
Single	0	Only show one document page.
Continuous	1	Show document pages continuously.

ZoomMode Enum

File
`wijmo.viewer.js`
Module
wijmo.viewer

Describes the supported zoom modes of FlexViewer.

Members

Name	Value	Description
Custom	0	Custom zoom mode. The actual zoom factor is determined by the value of the **`zoomFactor`** property.
PageWidth	1	Pages are zoomed in or out as necessary to fit the page width in the view panel.
WholePage	2	Pages are zoomed in or out as necessary to fit the whole page in the view panel.

wijmo.angular Module

File
`wijmo.angular.js`
Module
wijmo.angular

Contains AngularJS directives for the Wijmo controls.

The directives allow you to add Wijmo controls to **AngularJS** applications using simple markup in HTML pages.

You can use directives as regular HTML tags in the page markup. The tag name corresponds to the control name, prefixed with "wj-," and the attributes correspond to the names of control properties and events.

All control, property, and event names within directives follow the usual AngularJS convention of replacing camel-casing with hyphenated lower-case names.

AngularJS directive parameters come in three flavors, depending on the type of binding they use. The table below describes each one:

@
By value, or one-way binding. The attribute value is interpreted as a literal.
=
By reference, or two-way binding. The attribute value is interpreted as an expression.
&
Function binding. The attribute value is interpreted as a function call, including the parameters.

For more details on the different binding types, please see Dan Wahlin's blog on directives (http://weblogs.asp.net/dwahlin/creating-custom-angularjs-directives-part-2-isolate-scope). The documentation does not describe directive events because they are identical to the control events, and the binding mode is always the same (function binding).

To illustrate, here is the markup used to create a **ComboBox** control:

```
<wj-combo-box
  text="ctx.theCountry"
  items-source="ctx.countries"
  is-editable="true"
  selected-index-changed="ctx.selChanged(s, e)">
</wj-combo-box>
```

Notice that the **text** property of the **ComboBox** is bound to a controller variable called "ctx.theCountry." The binding goes two ways; changes in the control update the scope, and changes in the scope update the control. To initialize the **text** property with a string constant, enclose the attribute value in single quotes (for example, `text="'constant'"`).

Notice also that the **selected-index-changed** event is bound to a controller method called "selChanged," and that the binding includes the two event parameters (without the parameters, the method is not called). Whenever the control raises the event, the directive invokes the controller method.

All Wijmo Angular directives include an "initialized" event that is raised after the control has been added to the page and initialized. You can use this event to perform additional initialization in addition to setting properties in markup. For example:

```
<wj-flex-grid initialized="initGrid(s,e)">
</wj-flex-grid>


// controller
$scope.initGrid: function(s, e) {

  // assign a custom MergeManager to the grid
  s.mergeManager = new CustomMergeManager(s);

}
```

Classes

- WjAutoComplete
- WjBulletGraph
- WjCalendar
- WjCollectionViewNavigator
- WjCollectionViewPager
- WjColorPicker
- WjComboBox
- WjContextMenu
- WjFinancialChart
- WjFinancialChartSeries
- WjFlexChart
- WjFlexChartAnimation
- WjFlexChartAnnotation
- WjFlexChartAnnotationLayer
- WjFlexChartAtr
- WjFlexChartAxis
- WjFlexChartBollingerBands
- WjFlexChartBoxWhisker
- WjFlexChartCci
- WjFlexChartChartGestures
- WjFlexChartDataLabel
- WjFlexChartDataPoint
- WjFlexChartEnvelopes
- WjFlexChartErrorBar
- WjFlexChartFibonacci
- WjFlexChartFibonacciArcs
- WjFlexChartFibonacciFans
- WjFlexChartFibonacciTimeZones
- WjFlexChartLegend
- WjFlexChartLineMarker
- WjFlexChartMacd
- WjFlexChartMacdHistogram
- WjFlexChartMovingAverage
- WjFlexChartParametricFunctionSeries
- WjFlexChartRangeSelector
- WjFlexChartRsi
- WjFlexChartSeries
- WjFlexChartStochastic
- WjFlexChartTrendLine
- WjFlexChartWaterfall
- WjFlexChartWilliamsR
- WjFlexChartYFunctionSeries
- WjFlexGrid
- WjFlexGridCellTemplate
- WjFlexGridColumn
- WjFlexGridDetail
- WjFlexGridFilter
- WjFlexPie
- WjFlexPieDataLabel
- WjFlexRadar
- WjFlexRadarAxis
- WjFlexRadarSeries
- WjFlexSheet
- WjGroupPanel
- WjInputColor
- WjInputDate
- WjInputDateTime
- WjInputMask
- WjInputNumber
- WjInputTime
- WjItemTemplate
- WjLinearGauge
- WjListBox
- WjMenu
- WjMenuItem
- WjMenuSeparator
- WjMultiAutoComplete
- WjMultiRow
- WjMultiSelect
- WjPdfViewer
- WjPivotChart
- WjPivotGrid
- WjPivotPanel
- WjPopup
- WjRadialGauge
- WjRange
- WjReportViewer
- WjSheet
- WjSunburst
- WjTooltip
- WjTreeMap
- WjTreeView
- WjValidationError

Enums

- CellTemplateType

WjAutoComplete Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjComboBox
Derived Classes
WjMultiAutoComplete

AngularJS directive for the **AutoComplete** control.

Use the **wj-auto-complete** directive to add **AutoComplete** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an AutoComplete control:</p>
<wj-auto-complete
  text="theCountry"
  items-source="countries"
  is-editable="false"
  placeholder="country">
</wj-auto-complete>
```

The example below creates an **AutoComplete** control and binds it to a 'countries' array exposed by the controller. The **AutoComplete** searches for the country as the user types, and narrows down the list of countries that match the current input.

Example

Show me (http://jsfiddle.net/Wijmo5/37GHw)

The **wj-auto-complete** directive extends **WjComboBox** with the following attributes:

css-match
@ The name of the CSS class used to highlight parts of the content that match the search terms.
delay
@ The amount of delay in milliseconds between when a keystroke occurs and when the search is performed.
items-source-function
= A function that provides the items dynamically as the user types.
max-items
@ The maximum number of items to display in the dropdown.
min-length
@ The minimum input length to require before triggering autocomplete suggestions.

WjBulletGraph Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjLinearGauge

AngularJS directive for the **BulletGraph** control.

Use the **wj-bullet-graph** directive to add bullet graphs to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<wj-bullet-graph
  value="ctx.gauge.value"
  min="0" max="10"
  target="{{item.target}}"
  bad="{{item.target * .75}}"
  good="{{item.target * 1.25}}">
</wj-bullet-graph>
```

The **wj-bullet-graph** directive supports the following attributes:

control
= A reference to the BulletGraph control created by this directive.
direction
@ The **GaugeDirection** value indicating which direction the gauge fills as the value grows.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
target
@ The target value for the measure.
good
@ A reference value considered good for the measure.
bad
@ A reference value considered bad for the measure.
value
= The actual value of the measure.

The **wj-bullet-graph** directive may contain one or more **WjRange** directives.

Example

Show me (http://jsfiddle.net/Wijmo5/8uxb1vwf)

WjCalendar Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **Calendar** control.

Use the **wj-calendar** directive to add **Calendar** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a Calendar control:</p>
<wj-calendar
  value="theDate">
</wj-calendar>
```

Example

Show me (http://jsfiddle.net/Wijmo5/46PhD)

This example creates a **Calendar** control and binds it to a 'date' variable exposed by the controller. The range of dates that may be selected is limited by the **min** and **max** properties.

The **wj-calendar** directive supports the following attributes:

ng-model
`@` Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
`@` Specifies a control property that is bound to a scope using the **ng-model** directive.
control
`=` A reference to the **Calendar** control created by this directive.
display-month
`=` The month being displayed in the calendar.
first-day-of-week
`@` The first day of the week.
initialized
`&` This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
`=` A value indicating whether the binding has finished initializing the control with attribute values.
item-formatter
`=` The function used to customize the dates shown in the calendar.
max
`@` The latest valid date (string in the format "yyyy-MM-dd").
min

@ The earliest valid date (string in the format "yyyy-MM-dd").

month-view

@ A value indicating whether the control displays a month or the entire year.

show-header

@ A value indicating whether the control displays the header area.

value

= The date being edited.

got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

value-changed

& The **valueChanged** event handler.

If provided, the **min** and **max** attributes are strings in the format "yyyy-MM-dd." Technically, you can use any full date as defined in the W3C **[RFC 3339]**, which is also the format used with regular HTML5 input elements.

WjCollectionViewNavigator Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for an **ICollectionView** navigator element.

Use the **wj-collection-view-navigator** directive to add an element that allows users to navigate through the items in an **ICollectionView**. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
Here is a CollectionViewNavigator:</p>
<wj-collection-view-navigator
  cv="myCollectionView">
</wj-collection-view-navigator>
```

Example

Show me (http://jsfiddle.net/Wijmo5/s8tT4)

This example creates a CollectionView with 100,000 items and 20 items per page. It defines a navigator to select the current page, another to select the current item, and shows the data in a **FlexGrid**.

The **wj-collection-view-navigator** directive has a single attribute:

cv
= A reference to the **ICollectionView** object to navigate.

WjCollectionViewPager Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for an **ICollectionView** pager element.

Use the **wj-collection-view-pager** directive to add an element that allows users to navigate through the pages in a paged **ICollectionView**. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
Here is a CollectionViewPager:</p>
<wj-collection-view-pager
  cv="myCollectionView">
</wj-collection-view-pager>
```

Example

Show me (http://jsfiddle.net/Wijmo5/s8tT4)

This example creates a CollectionView with 100,000 items and 20 items per page. It defines a navigator to select the current page, another to select the current item, and shows the data in a **FlexGrid**.

The **wj-collection-view-pager** directive has a single attribute:

cv
= A reference to the paged **ICollectionView** object to navigate.

WjColorPicker Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **ColorPicker** control.

Use the **wj-color-picker** directive to add **ColorPicker** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a ColorPicker control:</p>
<wj-color-picker
  value="theColor"
  show-alpha-channel="false">
</wj-color-picker>
```

The **wj-color-picker** directive supports the following attributes:

ng-model
`@` Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
`@` Specifies a control property that is bound to a scope using the **ng-model** directive.
control
`=` A reference to the **ColorPicker** control created by this directive.
initialized
`&` This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
`=` A value indicating whether the binding has finished initializing the control with attribute values.
show-alpha-channel
`@` A value indicating whether the control displays the alpha channel (transparency) editor.
show-color-string
`@` A value indicating whether the control displays a string representation of the color being edited.
palette
`=` An array with ten color values to use as the palette.
value
`=` The color being edited.
got-focus
`&` The **gotFocus** event handler.
lost-focus
`&` The **lostFocus** event handler.
value-changed
`&` The **valueChanged** event handler.

WjComboBox Class

File
`wijmo.angular.js`
Module
wijmo.angular
Derived Classes
WjAutoComplete, **WjInputTime**, **WjMenu**, **WjMultiSelect**

AngularJS directive for the **ComboBox** control.

Use the **wj-combo-box** directive to add **ComboBox** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a ComboBox control:</p>
<wj-combo-box
  text="theCountry"
  items-source="countries"
  is-editable="false"
  placeholder="country">
</wj-combo-box>
```

The example below creates a **ComboBox** control and binds it to a 'countries' array exposed by the controller. The **ComboBox** searches for the country as the user types. The **isEditable** property is set to false, so the user is forced to select one of the items in the list.

Example

Show me (http://jsfiddle.net/Wijmo5/37GHw)

The **wj-combo-box** directive supports the following attributes:

ng-model
@ Binds the control's **selectedValue** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **ComboBox** control created by this directive.
display-member-path
@ The name of the property to use as the visual representation of the items.
is-content-html
@ A value indicating whether the drop-down list displays the items as plain text or as HTML.
is-dropped-down
@ A value indicating whether the drop down list is currently visible.
is-editable
@ A value indicating whether the user can enter values not present on the list.

initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
item-formatter
= A function used to customize the values shown in the drop-down list.
items-source
= An array or **ICollectionView** that contains items to show in the list.
max-drop-down-height
@ The maximum height of the drop-down list.
max-drop-down-width
@ The maximum width of the drop-down list.
placeholder
@ A string shown as a hint when the control is empty.
is-required
@ A value indicating whether to prevent null values.
show-drop-down-button
@ A value indicating whether the control displays a drop-down button.
selected-index
= The index of the currently selected item in the drop-down list.
selected-item
= The currently selected item in the drop-down list.
selected-value
= The value of the selected item, obtained using the **selected-value-path**.
selected-value-path
@ The name of the property used to get the **selected-value** from the **selected-item**.
text
= The text to show in the control.
is-dropped-down-changing
& The **isDroppedDownChanging** event handler.
is-dropped-down-changed
& The **isDroppedDownChanged** event handler.
selected-index-changed
& The **selectedIndexChanged** event handler.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.
text-changed
& The **textChanged** event handler.

WjContextMenu Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for context menus.

Use the **wj-context-menu** directive to add context menus to elements on the page. The wj-context-menu directive is based on the **wj-menu** directive; it displays a popup menu when the user performs a context menu request on an element (usually a right-click).

The wj-context-menu directive is specified as a parameter added to the element that the context menu applies to. The parameter value is a selector for the element that contains the menu. For example:

```
<!-- paragraph with a context menu -->
<p wj-context-menu="#idMenu" >
 This paragraph has a context menu.</p>

<!-- define the context menu (hidden and with an id) -->
<wj-menu id="idMenu" ng-show="false">
 <wj-menu-item cmd="cmdOpen" cmd-param ="1">Open...</wj-menu-item>
 <wj-menu-item cmd="cmdSave" cmd-param="2">Save </wj-menu-item>
 <wj-menu-item cmd="cmdSave" cmd-param="3">Save As...</wj-menu-item>
 <wj-menu-item cmd="cmdNew" cmd-param ="4">New...</wj-menu-item>
 <wj-menu-separator></wj-menu-separator>
 <wj-menu-item cmd="cmdExit" cmd-param="5">Exit</wj-menu-item>
</wj-menu >
```

WjFinancialChart Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart** control.

Use the **wj-financial-chart** directive to add financial charts to your AngularJS applications. Note that directive and parameter names must be formatted using lower-case letters with dashes instead of camel case.

The wj-financial-chart directive supports the following attributes:

binding
`@` The name of the property that contains Y values for the chart. You can override this at the series level.
binding-x
`@` The name of the property that contains X values for the chart. You can override this at the series level.
chart-type
`@` The default chart type to use in rendering series objects. You can override this at the series level. See **FinancialChartType**.
control
`=` A reference to the **FinancialChart** control that this directive creates.
footer
`@` The text to display in the chart footer (plain text).
footer-style
`=` The style to apply to the chart footer.
header
`@` The text to display in the chart header (plain text).
header-style
`=` The style to apply to the chart header.
initialized
`&` This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
`=` A value indicating whether the binding has finished initializing the control with attribute values.
interpolate-nulls
`@` The value indicating whether to interpolate or leave gaps when there are null values in the data.
item-formatter
`=` The formatter function that customizes the appearance of data points.
items-source
`=` An array or **ICollectionView** object that contains the data used to create the chart.
legend-toggle
`@` The value indicating whether clicking legend items toggles series visibility.
options
`=` Chart options that only apply to certain chart types. See **options** under **FinancialChart** for details.
palette
`=` An array that contains the default colors used for displaying each series.
plot-margin
`=` The number of pixels of space to leave between the edges of the control and the plot area, or CSS-style margins.

selection
= The series object that is selected.
selection-mode
@ The **SelectionMode** value indicating whether or what is selected when the user clicks a series.
symbol-size
@ The size of the symbols used to render data points in Scatter, LineSymbols, and SplineSymbols charts, in pixels. You can override this at the series level.
tooltip-content
@ The value to display in the **ChartTooltip** content property.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.
rendering
& The **rendering** event handler.
rendered
& The **rendered** event handler.
series-visibility-changed
& The **seriesVisibilityChanged** event handler.
selection-changed
& The **selectionChanged** event handler.

The wj-financial-chart directive may contain the following child directives: **WjFlexChartAxis**, **WjFlexChartSeries**, **WjFlexChartLegend** and **WjFlexChartDataLabel**.

WjFinancialChartSeries Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart FinancialSeries** object.

The **wj-financial-chart-series** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

axis-x
@ X-axis for the series.
axis-y
@ Y-axis for the series.
binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **FinancialChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any setting at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.

In most cases, the **wj-financial-chart-series** specifies the **name** and **binding** properties only. The remaining values are inherited from the parent **wj-financial-chart** directive.

WjFlexChart Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** control.

Use the **wj-flex-chart** directive to add charts to your AngularJS applications. Note that directive and parameter names must be formatted using lower-case letters with dashes instead of camel case. For example:

```
<p>Here is a FlexChart control:</p>
<wj-flex-chart
  style="height:300px"
  items-source="data"
  binding-x="country">
  <wj-flex-chart-axis
    wj-property="axisY"
    major-unit="5000">
  </wj-flex-chart-axis>
  <wj-flex-chart-series
    binding="sales"
    name="Sales">
  </wj-flex-chart-series>
  <wj-flex-chart-series
    binding="expenses"
    name="Expenses">
  </wj-flex-chart-series>
  <wj-flex-chart-series
    binding="downloads"
    name="Downloads"
    chart-type="LineSymbols">
  </wj-flex-chart-series>
</wj-flex-chart>
```

The example below creates a **FlexChart** control and binds it to a 'data' array exposed by the controller. The chart has three series objects, each corresponding to a property in the objects contained in the source array. The last series in the example uses the 'chart-type' attribute to override the default chart type used for the other series objects.

Example

Show me (http://jsfiddle.net/Wijmo5/QNb9X)

The wj-flex-chart directive supports the following attributes:

binding
@ The name of the property that contains Y values for the chart. You can override this at the series level.
binding-x
@ The name of the property that contains X values for the chart. You can override this at the series level.

chart-type
@ The default chart type to use in rendering series objects. You can override this at the series level. See **ChartType**.
control
= A reference to the **FlexChart** control that this directive creates.
footer
@ The text to display in the chart footer (plain text).
footer-style
= The style to apply to the chart footer.
header
@ The text to display in the chart header (plain text).
header-style
= The style to apply to the chart header.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
interpolate-nulls
@ The value indicating whether to interpolate or leave gaps when there are null values in the data.
item-formatter
= The formatter function that customizes the appearance of data points.
items-source
= An array or **ICollectionView** object that contains the data used to create the chart.
legend-toggle
@ The value indicating whether clicking legend items toggles series visibility.
options
= Chart options that only apply to certain chart types. See **options** under **FlexChart** for details.
palette
= An array that contains the default colors used for displaying each series.
plot-margin
= The number of pixels of space to leave between the edges of the control and the plot area, or CSS-style margins.
rotated
@ The value indicating whether to flip the axes so that X is vertical and Y is horizontal.
selection
= The series object that is selected.
selection-mode
@ The **SelectionMode** value indicating whether or what is selected when the user clicks a series.
stacking
@ The **Stacking** value indicating whether or how series objects are stacked or plotted independently.
symbol-size
@ The size of the symbols used to render data points in Scatter, LineSymbols, and SplineSymbols charts, in pixels. You can override this at the series level.
tooltip-content
@ The value to display in the **ChartTooltip** content property.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.
rendering
& The **rendering** event handler.
rendered

& The **rendered** event handler.

series-visibility-changed

& The **seriesVisibilityChanged** event handler.

selection-changed

& The **selectionChanged** event handler.

The wj-flex-chart directive may contain the following child directives: **WjFlexChartAxis**, **WjFlexChartSeries**, **WjFlexChartLegend** and **WjFlexChartDataLabel**.

WjFlexChartAnimation Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart ChartAnimation** object.

The **wj-flex-chart-animation** directive must be contained in a **WjFlexChart** or **WjFlexPie** or **WjFinancialChart** directive. It supports the following attributes:

animation-mode
@ The value indicating whether the plot points animate one at a time, series by series, or all at once.
easing
@ The value indicating the easing function applied to the animation.
duration
@ The value indicating the length of entire animation in milliseconds.
axis-animation
@ The value indicating whether the axis animation is enabled.

WjFlexChartAnnotation Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the annotations.

The **wj-flex-chart-annotation** directive must be contained in a **WjFlexChartAnnotationLayer** directive.

The **wj-flex-chart-annotation** directive is used to represent all types of possible annotation shapes like **Circle**, **Rectangle**, **Polygon** and so on. The type of annotation shape is specified in the directive's **type** attribute.

The directive supports the following attributes:

type
@ The class name of the annotation shape represented by the directive. The possible values are **Circle**, **Ellipse**, **Image**, **Line**, **Polygon**, **Rectangle**, **Square**, **Text**.
attachment
@ An **AnnotationAttachment** value defining the attachment of the annotation.
content
@ The text of the **Circle**, **Ellipse**, **Image**, **Line**, **Polygon**, **Rectangle** or **Square** annotation.
end
@ The end point of the **Line** annotation.
height
@ The height of the **Ellipse**, **Image** or **Rectangle** annotation.
href
@ The href of the **Image** annotation.
is-visible
@ The visibility of the annotation.
length
@ The length of the **Square** annotation.
name
@ The name of the annotation.
offset
@ The offset of the annotation.
point
@ The point of the annotation, the coordinate space of the point depends on the **attachment** property value. The property works for **Circle**, **Ellipse**, **Image**, **Rectangle**, **Square** and **Text** annotation.
point-index
@ The index of the data point in the specified series where the annotation is attached to.
position
@ An **AnnotationPosition** value defining the position of the annotation relative to the **point**.
radius
@ The radius of the **Circle** annotation.
series-index
@ The index of the data series where the annotation is attached to.
start
@ The start point of the **Line** annotation.

style

@ The style of the annotation.

text

@ The text of the **Text** annotation.

tooltip

@ The tooltip of the annotation.

width

@ The width of the **Ellipse**, **Image** or **Rectangle** annotation.

WjFlexChartAnnotationLayer Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart AnnotationLayer** object.

The **wj-flex-chart-annotation-layer** directive must be contained in a **WjFlexChart** directive or **WjFinancialChart** directive.

WjFlexChartAtr Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart ATR** object.

The **wj-flex-chart-atr** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
period
@ The period for the average true range calculation.

WjFlexChartAxis Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart Axis** object.

The **wj-flex-chart-axis** directive must be contained in a **WjFlexChart** directive or **WjFinancialChart** directive. It supports the following attributes:

wj-property
@ Defines the **FlexChart** property name, axis-x or axis-y, to initialize with the directive.
axis-line
@ The value indicating whether the axis line is visible.
binding
@ Gets or sets the comma-separated property names for the **itemsSource** property to use in axis labels. The first name specifies the value on the axis, the second represents the corresponding axis label. The default value is 'value,text'.
format
@ The format string used for the axis labels (see **Globalize**).
item-formatter
= The formatter function that customizes the appearance of axis labels.
items-source
= The items source for the axis labels.
labels
@ The value indicating whether the axis labels are visible.
label-angle
@ The rotation angle of axis labels in degrees.
label-align
@ The alignment of axis labels.
label-padding
@ The padding of axis labels.
major-grid
@ The value indicating whether the axis includes grid lines.
major-tick-marks
@ Defines the appearance of tick marks on the axis (see **TickMark**).
major-unit
@ The number of units between axis labels.
max
@ The minimum value shown on the axis.
min
@ The maximum value shown on the axis.
minor-grid
@ The value indicating whether the axis includes minor grid lines.
minor-tick-marks
@ Defines the appearance of minor tick marks on the axis (see **TickMark**).
minor-unit
@ The number of units between minor axis ticks.

origin

@ The axis origin.

overlappingLabels

@ The **OverlappingLabels** value indicating how to handle the overlapping axis labels.

position

@ The **Position** value indicating the position of the axis.

reversed

@ The value indicating whether the axis is reversed (top to bottom or right to left).

title

@ The title text shown next to the axis.

WjFlexChartBollingerBands Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart BollingerBands** object.

The **wj-flex-chart-bollinger-bands** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
period
@ The period for the Bollinger Bands calculation.
multiplier/dt>
@ The standard deviation multiplier for the Bollinger Bands calculation.

WjFlexChartBoxWhisker Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** and **FinancialChart BoxWhisker** object.

The **wj-flex-chart-box-whisker** directive must be contained in a **WjFlexChart** or **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
quartile-calculation
@ The value that specifies the quartile calculation for the Box&Whisker chart.
group-width
@ The value that determines the group width as a percentage for the Box&Whisker chart.
gap-width
@ The value that determines the gap width as a percentage for the Box&Whisker chart.
show-mean-line
@ The value that determines whether to show the mean line for the Box&Whisker chart.
mean-line-style
@ The value that specifies the style for the mean line.
show-mean-marker
@ The value that determines whether to show the mean marker for the Box&Whisker chart.
mean-marker-style
@ The value that specifies the style for the mean marker.
show-inner-points
@ The value that determines whether to show the inner points for the Box&Whisker chart.
show-outliers
@ The value that determines whether to show the outliers for the Box&Whisker chart.

WjFlexChartCci Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart CCI** object.

The **wj-flex-chart-cci** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
period
@ The period for the commodity channel index calculation.

WjFlexChartChartGestures Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart ChartGestures** object.

The **wj-flex-chart-gestures** directive must be contained in a **WjFlexChart** directive or **WjFinancialChart** directive. It supports the following attributes:

mouse-action
@ The value indicating mouse action is zooming or panning.
interactive-axes
@ The value indicating which axis is interactive.
enable
@ The value indicating the gestures action is enabled or not.
scale-x
@ The value indicating axisX initial range between Min and Max.
scale-y
@ The value indicating axisY initial range between Min and Max.
pos-x
@ The value indicating initial position on the axisX.
pos-y
@ The value indicating initial position on the axisY.

WjFlexChartDataLabel Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart DataLabel** object.

The **wj-flex-chart-data-label** directive must be contained in a **WjFlexChart** directive. It supports the following attributes:

content
= A string or function that gets or sets the content of the data labels.
border
@ Gets or sets a value indicating whether the data labels have borders.
position
@ The **LabelPosition** value indicating the position of the data labels.

WjFlexChartDataPoint Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart DataPoint** object.

The **wj-flex-chart-data-point** directive must be contained in a **WjFlexChartAnnotation** directive. The property of the parent directive's object where **wj-flex-data-point** should assign a value is specified in the **wj-property** attribute.

The directive supports the following attributes:

wj-property
@ The name of the parent directive object's property where the **DataPoint** will be assigned.
x
@ x coordinate, can be a numeric or date value.
y
@ y coordinate, can be a numeric or date value.

WjFlexChartEnvelopes Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart Envelopes** object.

The **wj-flex-chart-envelopes** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
period
@ The period for the moving average envelopes calculation.
size/dt>
@ The size of the moving average envelopes.
type/dt>
@ The **MovingAverageType** of the moving average to be used for the envelopes.

WjFlexChartErrorBar Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjFlexChartSeries

AngularJS directive for the **FlexChart ErrorBar** object.

The **wj-flex-chart-error-bar** directive must be contained in a **WjFlexChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
error-bar-style
@ The value that specifies the ErrorBar style.
value
@ The value that specifies the error value of the series.
error-amount
@ The value that specifies the error amount of the series.
end-style
@ The value that specifies the end style of the series.
direction
@ The value that specifies the direction of the series.

WjFlexChartFibonacci Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart Fibonacci** object.

The **wj-flex-chart-fibonacci** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
high
@ The high value of **Fibonacci** tool.
labelPosition
@ The label position for levels in **Fibonacci** tool.
levels
@ The levels value of **Fibonacci** tool.
low
@ The low value of **Fibonacci** tool.
minX
@ The x minimum value of **Fibonacci** tool.
maxX
@ The x maximum value of **Fibonacci** tool.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
uptrend
@ The value indicating whether to create uptrending **Fibonacci** tool.

WjFlexChartFibonacciArcs Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart FibonacciArcs** object.

The **wj-flex-chart-fibonacci-arcs** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
labelPosition
@ The **LabelPosition** for levels in **FibonacciArcs** tool.
levels
@ The levels value of **FibonacciArcs** tool.
start-x
@ The starting X value of **FibonacciArcs** tool.
end-x
@ The ending X value of **FibonacciArcs** tool.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.

WjFlexChartFibonacciFans Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart FibonacciFans** object.

The **wj-flex-chart-fibonacci-fans** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
labelPosition
@ The **LabelPosition** for levels in **FibonacciFans** tool.
levels
@ The levels value of **FibonacciFans** tool.
start
@ The starting **DataPoint** of **FibonacciFans** tool.
end
@ The ending **DataPoint** of **FibonacciFans** tool.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.

WjFlexChartFibonacciTimeZones Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart FibonacciTimeZones** object.

The **wj-flex-chart-fibonacci-time-zones** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
labelPosition
@ The **LabelPosition** for levels in **FibonacciTimeZones** tool.
levels
@ The levels value of **FibonacciTimeZones** tool.
startX
@ The starting X value of **FibonacciTimeZones** tool.
endX
@ The ending X value of **FibonacciTimeZones** tool.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.

WjFlexChartLegend Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart Legend** object.

The **wj-flex-chart-legend** directive must be contained in a **WjFlexChart** directive, **WjFlexPie** directive or **WjFinancialChart** directive. It supports the following attributes:

position
`@` The **Position** value indicating the position of the legend.

The example below shows how you can use the wj-flex-chart-legend directive to change the position of the chart legend:

```
<wj-flex-chart
  items-source="data"
  binding-x="country">
  <wj-flex-chart-axis
      wj-property="axisY"
      major-unit="5000">
    </wj-flex-chart-axis>
    <wj-flex-chart-series
      binding="sales"
      name="Sales">
    </wj-flex-chart-series>
  <wj-flex-chart-legend
    position="Bottom">
  </wj-flex-chart-legend>
</wj-flex-chart>
```

WjFlexChartLineMarker Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart LineMarker** object.

The **wj-flex-line-marker** directive must be contained in a **WjFlexChart** directive or **WjFinancialChart** directive. It supports the following attributes:

is-visible
@ The value indicating whether the LineMarker is visible.
series-index
@ The index of the series in the chart in which the LineMarker appears.
horizontal-position
@ The horizontal position of the LineMarker relative to the plot area.
content
@ The function that allows you to customize the text content of the LineMarker.
vertical-position
@ The vertical position of the LineMarker relative to the plot area.
alignment
@ The **LineMarkerAlignment** value indicating the alignment of the LineMarker content.
lines
@ The **LineMarkerLines** value indicating the appearance of the LineMarker's lines.
interaction
@ The **LineMarkerInteraction** value indicating the interaction mode of the LineMarker.
drag-threshold
@ The maximum distance from the horizontal or vertical line that you can drag the marker.
drag-content
@ The value indicating whether you can drag the content of the marker when the interaction mode is "Drag".
drag-lines
@ The value indicating whether the lines are linked when you drag the horizontal or vertical line when the interaction mode is "Drag".

WjFlexChartMacd Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart Macd** object.

The **wj-flex-chart-macd** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
styles/dt>
The styles for the MACD and Signal lines.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
fast-period
@ The fast moving average period for the MACD calculation.
slow-period
@ The slow moving average period for the MACD calculation.
signal-smoothing-period/dt>
@ The smoothing period for the MACD calculation.

WjFlexChartMacdHistogram Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart MacdHistogram** object.

The **wj-flex-chart-macd-histogram** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
fast-period
@ The fast moving average period for the MACD calculation.
slow-period
@ The slow moving average period for the MACD calculation.
signal-smoothing-period/dt>
@ The smoothing period for the MACD calculation.

WjFlexChartMovingAverage Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** and **FinancialChart MovingAverage** object.

The **wj-flex-chart-moving-average** directive must be contained in a **WjFlexChart** or **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **ChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
type
@ The **MovingAverageType** value for the moving average series.
period
@ The period for the moving average calculation.

WjFlexChartParametricFunctionSeries Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** and **FinancialChart WjFlexChartParametricFunctionSeries** object.

The **wj-flex-chart-parametric-function-series** directive must be contained in a **WjFlexChart** or **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **ChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
sample-count
@ The sample count for the calculation.
min
@ The minimum value of the parameter for calculating a function.
max
@ The maximum value of the parameter for calculating a function.
x-func
@ The function used to calculate the x value.
y-func
@ The function used to calculate the y value.

WjFlexChartRangeSelector Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart RangeSelector** object.

The **wj-flex-chart-range-selector** directive must be contained in a **WjFlexChart** directive or **WjFinancialChart** directive. It supports the following attributes:

is-visible
`@` The value indicating whether the RangeSelector is visible.
min
`@` The minimum value of the range.
max
`@` The maximum value of the range.
orientation
`@` The orientation of the RangeSelector.
seamless
`@` The value indicating whether the minimal and maximal handler will move seamlessly.
min-scale
`@` the valid minimum range of the RangeSelector.
max-scale
`@` the valid maximum range of the RangeSelector.

WjFlexChartRsi Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart RSI** object.

The **wj-flex-chart-rsi** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
period
@ The period for the relative strength index calculation.

WjFlexChartSeries Class

File
`wijmo.angular.js`
Module
wijmo.angular
Derived Classes
WjFlexChartErrorBar

AngularJS directive for the **FlexChart Series** object.

The **wj-flex-chart-series** directive must be contained in a **WjFlexChart** directive. It supports the following attributes:

axis-x
@ X-axis for the series.
axis-y
@ Y-axis for the series.
binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **ChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any settings at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any settings at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.

In most cases, the **wj-flex-chart-series** specifies only the **name** and **binding** properties. The remaining values are inherited from the parent **wj-flex-chart** directive.

WjFlexChartStochastic Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart Stochastic** object.

The **wj-flex-chart-stochastic** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
styles/dt>
The styles for the %K and %D lines.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
k-period
@ The period for the %K calculation.
d-period
@ The period for the %D calculation.
smoothing-period
@ The smoothing period for the %K calculation.

WjFlexChartTrendLine Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** and **FinancialChart TrendLine** object.

The **wj-flex-chart-trend-line** directive must be contained in a **WjFlexChart** or **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **ChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any setting at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
sample-count
@ The sample count for the calculation.
fit-type
@ The **TrendLineFitType** value for the trend line.
order
@ The number of terms in a polynomial or fourier equation.

WjFlexChartWaterfall Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** and **FinancialChart Waterfall** object.

The **wj-flex-chart-waterfall** directive must be contained in a **WjFlexChart** or **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
relative-data
@ The value that determines whether the given data is relative.
start
@ The value of the start bar.
start-label
@ The label of the start bar.
show-total
@ The value that determines whether the show the total bar.
total-label
@ The label of the total bar.
show-intermediate-total
@ The value that determines whether to show the intermediate total bar.
intermediate-total-positions
@ The value that contains the index for positions of the intermediate total bar.
intermediate-total-labels
@ The value that contains the label of the intermediate total bar.
connector-lines
@ The value that determines whether to show connector lines.
styles
@ The value of the waterfall styles.

WjFlexChartWilliamsR Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart WilliamsR** object.

The **wj-flex-chart-williams-r** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
period
@ The period for the Williams %R calculation.

WjFlexChartYFunctionSeries Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexChart** and **FinancialChart YFunctionSeries** object.

The **wj-flex-chart-y-function-series** directive must be contained in a **WjFlexChart** or **WjFinancialChart** directive. It supports the following attributes:

binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **ChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any set at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.
sample-count
@ The sample count for the calculation.
min
@ The minimum value of the parameter for calculating a function.
max
@ The maximum value of the parameter for calculating a function.
func
@ The function used to calculate Y value.

WjFlexGrid Class

File
`wijmo.angular.js`
Module
wijmo.angular
Derived Classes
WjFlexSheet, **WjMultiRow**, **WjPivotGrid**

AngularJS directive for the **FlexGrid** control.

Use the **wj-flex-grid** directive to add grids to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a FlexGrid control:</p>
<wj-flex-grid items-source="data">
  <wj-flex-grid-column
    header="Country"
    binding="country">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Sales"
    binding="sales">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Expenses"
    binding="expenses">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Downloads"
    binding="downloads">
  </wj-flex-grid-column>
</wj-flex-grid>
```

The example below creates a FlexGrid control and binds it to a 'data' array exposed by the controller. The grid has three columns, each corresponding to a property of the objects contained in the source array.

Example

Show me (http://jsfiddle.net/Wijmo5/QNb9X)

The **wj-flex-grid** directive supports the following attributes:

allow-add-new
@ A value indicating whether to show a new row template so users can add items to the source collection.
allow-delete
@ A value indicating whether the grid deletes the selected rows when the Delete key is pressed.
allow-dragging
@ An **AllowDragging** value indicating whether and how the user can drag rows and columns with the mouse.
allow-merging
@ An **AllowMerging** value indicating which parts of the grid provide cell merging.

allow-resizing
@ An **AllowResizing** value indicating whether users are allowed to resize rows and columns with the mouse.
allow-sorting
@ A boolean value indicating whether users can sort columns by clicking the column headers.
auto-generate-columns
@ A boolean value indicating whether the grid generates columns automatically based on the **items-source**.
child-items-path
@ The name of the property used to generate child rows in hierarchical grids (or an array of property names if items at different hierarchical levels use different names for their child items).
control
= A reference to the **FlexGrid** control created by this directive.
defer-resizing
= A boolean value indicating whether row and column resizing should be deferred until the user releases the mouse button.
frozen-columns
@ The number of frozen (non-scrollable) columns in the grid.
frozen-rows
@ The number of frozen (non-scrollable) rows in the grid.
group-header-format
@ The format string used to create the group header content.
headers-visibility
= A **HeadersVisibility** value indicating whether the row and column headers are visible.
ime-enabled
@ Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
item-formatter
= A function that customizes cells on this grid.
items-source
= An array or **ICollectionView** object that contains the items shown on the grid.
is-read-only
@ A boolean value indicating whether the user is prevented from editing grid cells by typing into them.
merge-manager
= A **MergeManager** object that specifies the merged extent of the specified cell.
selection-mode
@ A **SelectionMode** value indicating whether and how the user can select cells.
show-groups
@ A boolean value indicating whether to insert group rows to delimit data groups.
show-sort
@ A boolean value indicating whether to display sort indicators in the column headers.
sort-row-index
@ A number specifying the index of row in the column header panel that shows and changes the current sort.
tree-indent
@ The indentation, in pixels, used to offset row groups of different levels.
beginning-edit
& Handler for the **beginningEdit** event.
cell-edit-ended
& Handler for the **cellEditEnded** event.
cell-edit-ending

& Handler for the **cellEditEnding** event.
prepare-cell-for-edit
& Handler for the **prepareCellForEdit** event.
resizing-column
& Handler for the **resizingColumn** event.
resized-column
& Handler for the **resizedColumn** event.
dragged-column
& Handler for the **draggedColumn** event.
dragging-column
& Handler for the **draggingColumn** event.
sorted-column
& Handler for the **sortedColumn** event.
sorting-column
& Handler for the **sortingColumn** event.
deleting-row
& Handler for the **deletingRow** event.
dragging-row
& Handler for the **draggingRow** event.
dragged-row
& Handler for the **draggedRow** event.
resizing-row
& Handler for the **resizingRow** event.
resized-row
& Handler for the **resizedRow** event.
row-added
& Handler for the **rowAdded** event.
row-edit-ended
& Handler for the **rowEditEnded** event.
row-edit-ending
& Handler for the **rowEditEnding** event.
loaded-rows
& Handler for the **loadedRows** event.
loading-rows
& Handler for the **loadingRows** event.
group-collapsed-changed
& Handler for the **groupCollapsedChanged** event.
group-collapsed-changing
& Handler for the **groupCollapsedChanging** event.
items-source-changed
& Handler for the **itemsSourceChanged** event.
selection-changing
& Handler for the **selectionChanging** event.
selection-changed
& Handler for the **selectionChanged** event.
got-focus
& Handler for the **gotFocus** event.
lost-focus
& Handler for the **lostFocus** event.

scroll-position-changed
& Handler for the **scrollPositionChanged** event.

The **wj-flex-grid** directive may contain **WjFlexGridColumn**, **WjFlexGridCellTemplate** and **WjFlexGridDetail** child directives.

WjFlexGridCellTemplate Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexGrid** cell templates.

The **wj-flex-grid-cell-template** directive defines a template for a certain cell type in **FlexGrid**, and must contain a **cell-type** attribute that specifies the **CellTemplateType**. Depending on the template's cell type, the **wj-flex-grid-cell-template** directive must be a child of either **WjFlexGrid** or **WjFlexGridColumn** directives.

Column-specific cell templates must be contained in **wj-flex-grid-column** directives, and cells that are not column-specific (like row header or top left cells) must be contained in the **wj-flex-grid directive**.

In addition to an HTML fragment, **wj-flex-grid-cell-template** directives may contain an **ng-style** or **ng-class** attribute that provides conditional formatting for cells.

Both the **ng-style/ng-class** attributes and the HTML fragment can use the **$col**, **$row** and **$item** template variables that refer to the **Column**, **Row** and **Row.dataItem** objects pertaining to the cell.

For cell types like **Group** and **CellEdit**, an additional **$value** variable containing an unformatted cell value is provided. For example, here is a FlexGrid control with templates for row headers and for the Country column's regular and column header cells:

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeader">
    {{$row.index}}
  </wj-flex-grid-cell-template>
  <wj-flex-grid-cell-template cell-type="RowHeaderEdit">
    ...
  </wj-flex-grid-cell-template>

  <wj-flex-grid-column header="Country" binding="country">
    <wj-flex-grid-cell-template cell-type="ColumnHeader">
      <img ng-src="resources/globe.png" />
        {{$col.header}}
      </wj-flex-grid-cell-template>
      <wj-flex-grid-cell-template cell-type="Cell">
        <img ng-src="resources/{{$item.country}}.png" />
        {{$item.country}}
      </wj-flex-grid-cell-template>
    </wj-flex-grid-column>
  <wj-flex-grid-column header="Sales" binding="sales"></wj-flex-grid-column>
</wj-flex-grid>
```

For more detailed information on specific cell type templates refer to the documentation for the **CellTemplateType** enumeration.

Note that the **wj-flex-grid-column** directive may also contain arbitrary content that is treated as a template for a regular data cell (*cell-type*="*Cell*"). But if a **wj-flex-grid-cell-template** directive exists and is set to *cell-type*="*Cell*" under the **wj-flex-grid-column** directive, it takes priority and overrides the arbitrary content.

The **wj-flex-grid-cell-template** directive supports the following attributes:

cell-type

@ The **CellTemplateType** value defining the type of cell the template applies to.

cell-overflow

@ Defines the **style.overflow** property value for cells.

force-full-edit

@ For cell edit templates, indicates whether cell editing forcibly starts in full edit mode, regardless of how the editing was initiated. In full edit mode pressing cursor keys don't finish editing. Defaults to true.

The **cell-type** attribute takes any of the following enumerated values:

Cell

Defines a regular (data) cell template. Must be a child of the **WjFlexGridColumn** directive. For example, this cell template shows flags in the Country column's cells:

```
<wj-flex-grid-column header="Country" binding="country">
  <wj-flex-grid-cell-template cell-type="Cell">
    <img ng-src="resources/{{$item.country}}.png" />
    {{$item.country}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

For a hierarchical **FlexGrid** (that is, one with the **childItemsPath** property specified), if no **Group** template is provided, non-header cells in group rows in this **Column** also use this template.

CellEdit

Defines a template for a cell in edit mode. Must be a child of the **WjFlexGridColumn** directive. This cell type has an additional **$value** property available for binding. It contains the original cell value before editing, and the updated value after editing. For example, here is a template that uses the Wijmo **InputNumber** control as an editor for the "Sales" column:

```
<wj-flex-grid-column header="Sales" binding="sales">
  <wj-flex-grid-cell-template cell-type="CellEdit">
    <wj-input-number value="$value" step="1"></wj-input-number>
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

ColumnHeader

Defines a template for a column header cell. Must be a child of the **WjFlexGridColumn** directive. For example, this template adds an image to the header of the "Country" column:

```
<wj-flex-grid-column header="Country" binding="country">
  <wj-flex-grid-cell-template cell-type="ColumnHeader">
    <img ng-src="resources/globe.png" />
    {{$col.header}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

RowHeader

Defines a template for a row header cell. Must be a child of the **WjFlexGrid** directive. For example, this template shows row indices in the row headers:

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeader">
    {{$row.index}}
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

Note that this template is applied to a row header cell, even if it is in a row that is in edit mode. In order to provide an edit-mode version of a row header cell with alternate content, define the **RowHeaderEdit** template.

RowHeaderEdit

Defines a template for a row header cell in edit mode. Must be a child of the **WjFlexGrid** directive. For example, this template shows dots in the header of rows being edited:

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeaderEdit">
      ...
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

To add the standard edit-mode indicator to cells where the **RowHeader** template applies, use the following **RowHeaderEdit** template:

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="RowHeaderEdit">
    {{&#x270e;}}
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

TopLeft

Defines a template for the top left cell. Must be a child of the **WjFlexGrid** directive. For example, this template shows a down/right glyph in the top-left cell of the grid:

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="TopLeft">
    <span class="wj-glyph-down-right"></span>
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

GroupHeader

Defines a template for a group header cell in a **GroupRow**, Must be a child of the **WjFlexGridColumn** directive.

The **$row** variable contains an instance of the **GroupRow** class. If the grouping comes from the a **CollectionView**, the **$item** variable references the **CollectionViewGroup** object.

For example, this template uses a checkbox element as an expand/collapse toggle:

```
<wj-flex-grid-column header="Country" binding="country">
  <wj-flex-grid-cell-template cell-type="GroupHeader">
    <input type="checkbox" ng-model="$row.isCollapsed"/>
    {{$item.name}} ({{$item.items.length}} items)
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

Group

Defines a template for a regular cell (not a group header) in a **GroupRow**. Must be a child of the **WjFlexGridColumn** directive. This cell type has an additional **$value** varible available for binding. In cases where columns have the **aggregate** property specified, it contains the unformatted aggregate value.

For example, this template shows an aggregate's value and kind for group row cells in the "Sales" column:

```
<wj-flex-grid-column header="Sales" binding="sales" aggregate="Avg">
  <wj-flex-grid-cell-template cell-type="Group">
    Average: {{$value | number:2}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

ColumnFooter

Defines a template for a regular cell in a **columnFooters** panel. Must be a child of the **WjFlexGridColumn** directive. This cell type has an additional **$value** property available for binding that contains a cell value.

For example, this template shows aggregate's value and kind for a footer cell in the "Sales" column:

```
<wj-flex-grid-column header="Sales" binding="sales" aggregate="Avg">
  <wj-flex-grid-cell-template cell-type="ColumnFooter">
    Average: {{$value | number:2}}
  </wj-flex-grid-cell-template>
</wj-flex-grid-column>
```

BottomLeft

Defines a template for the bottom left cells (at the intersection of the row header and column footer cells). Must be a child of the **WjFlexGrid** directive. For example, this template shows a sigma glyph in the bottom-left cell of the grid:

```
<wj-flex-grid items-source="data">
  <wj-flex-grid-cell-template cell-type="BottomLeft">
   &#931;
  </wj-flex-grid-cell-template>
</wj-flex-grid>
```

WjFlexGridColumn Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **Column** object.

The **wj-flex-grid-column** directive must be contained in a **WjFlexGrid** directive. It supports the following attributes:

aggregate
@ The **Aggregate** object to display in the group header rows for this column.
align
@ The string value that sets the horizontal alignment of items in the column to left, right, or center.
allow-dragging
@ The value indicating whether the user can move the column to a new position with the mouse.
allow-sorting
@ The value indicating whether the user can sort the column by clicking its header.
allow-resizing
@ The value indicating whether the user can resize the column with the mouse.
allow-merging
@ The value indicating whether the user can merge cells in the column.
binding
@ The name of the property to which the column is bound.
css-class
@ The name of a CSS class to use when rendering the column.
data-map
= The **DataMap** object to use to convert raw values into display values for the column.
data-type
@ The enumerated **DataType** value that indicates the type of value stored in the column.
format
@ The format string to use to convert raw values into display values for the column (see **Globalize**).
header
@ The string to display in the column header.
input-type
@ The type attribute to specify the input element used to edit values in the column. The default is "tel" for numeric columns, and "text" for all other non-Boolean columns.
is-content-html
@ The value indicating whether cells in the column contain HTML content rather than plain text.
is-read-only
@ The value indicating whether the user is prevented from editing values in the column.
is-selected
@ The value indicating whether the column is selected.
mask
@ The mask string used to edit values in the column.
max-width
@ The maximum width for the column.
min-width

@ The minimum width for the column.

name

@ The column name. You can use it to retrieve the column.

is-required

@ The value indicating whether the column must contain non-null values.

show-drop-down

@ The value indicating whether to show drop-down buttons for editing based on the column's **DataMap**.

visible

@ The value indicating whether the column is visible.

width

@ The width of the column in pixels or as a star value.

word-wrap

@ The value indicating whether cells in the column wrap their content.

Any html content within the **wj-flex-grid-column** directive is treated as a template for the cells in that column. The template is applied only to regular cells. If you wish to apply templates to specific cell types such as column or group headers, then please see the **WjFlexGridCellTemplate** directive.

The following example creates two columns with a template and a conditional style:

Example

Show me (http://jsfiddle.net/Wijmo5/5L423)

The **wj-flex-grid-column** directive may contain **WjFlexGridCellTemplate** child directives.

WjFlexGridDetail Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for **FlexGrid DetailRow** templates.

The **wj-flex-grid-detail** directive must be contained in a **wj-flex-grid** directive.

The **wj-flex-grid-detail** directive represents a **FlexGridDetailProvider** object that maintains detail rows visibility, with detail rows content defined as an arbitrary HTML fragment within the directive tag. The fragment may contain AngularJS bindings and directives. In addition to any properties available in a controller, the local **$row** and **$item** template variables can be used in AngularJS bindings that refer to the detail row's parent **Row** and **Row.dataItem** objects. For example:

```
<p>Here is a detail row with a nested FlexGrid:</p>

<wj-flex-grid
  items-source="categories">
  <wj-flex-grid-column header="Name" binding="CategoryName"></wj-flex-grid-column>
  <wj-flex-grid-column header="Description" binding="Description" width="*"></wj-flex-grid-column>
  <wj-flex-grid-detail max-height="250" detail-visibility-mode="detailMode">
    <wj-flex-grid
      items-source="getProducts($item.CategoryID)"
      headers-visibility="Column">
    </wj-flex-grid>
  </wj-flex-grid-detail>
</wj-flex-grid>
```

A reference to a **FlexGridDetailProvider** object represented by the **wj-flex-grid-detail** directive can be retrieved in a usual way by binding to the directive's **control** property. This makes all the API provided by **FlexGridDetailProvider** available for usage in the template, giving you total control over the user experience. The following example adds a custom show/hide toggle to the Name column cells, and a Hide Detail button to the detail row. These elements call the **FlexGridDetailProvider**, **hideDetail** and **showDetail** methods in their **ng-click** bindings to implement the custom show/hide logic:

```
<p>Here is a FlexGrid with custom show/hide detail elements:</p>

<wj-flex-grid
  items-source="categories"
  headers-visibility="Column"
  selection-mode="Row">
  <wj-flex-grid-column header="Name" binding="CategoryName" is-read-only="true" width="200">
    <img ng-show="dp.isDetailVisible($row)" ng-click="dp.hideDetail($row)" src="resources/hide.png" />
    <img ng-hide="dp.isDetailVisible($row)" ng-click="dp.showDetail($row, true)" src="resources/show.png" />
    {{$item.CategoryName}}
  </wj-flex-grid-column>
  <wj-flex-grid-column header="Description" binding="Description" width="2*"></wj-flex-grid-column>
  <wj-flex-grid-detail control="dp" detail-visibility-mode="Code">
    <div style="padding:12px;background-color:#cee6f7">
      ID: <b>{{$item.CategoryID}}</b><br />
      Name: <b>{{$item.CategoryName}}</b><br />
      Description: <b>{{$item.Description}}</b><br />
      <button class="btn btn-default" ng-click="dp.hideDetail($row)">Hide Detail</button>
    </div>
  </wj-flex-grid-detail>
</wj-flex-grid>
```

The **wj-flex-grid-detail** directive supports the following attributes:

control
= A reference to the **FlexGridDetailProvider** object created by this directive.
detail-visibility-mode
@ A **DetailVisibilityMode** value that determines when to display the row details.
max-height
@ The maximum height of the detail rows, in pixels.
row-has-detail
= The callback function that determines whether a row has details.

WjFlexGridFilter Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexGridFilter** object.

The **wj-flex-grid-filter** directive must be contained in a **WjFlexGrid** directive. For example:

```
<p>Here is a FlexGrid control with column filters:</p>
<wj-flex-grid items-source="data">
  <wj-flex-grid-filter filter-columns="['country', 'expenses']"></wj-flex-grid-filter>

  <wj-flex-grid-column
    header="Country"
    binding="country">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Sales"
    binding="sales">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Expenses"
    binding="expenses">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Downloads"
    binding="downloads">
  </wj-flex-grid-column>
 </wj-flex-grid>
```

The **wj-flex-grid-filter** directive supports the following attributes:

filter-columns
`=` An array containing the names or bindings of the columns to filter.
show-filter-icons
`@` The value indicating whether filter editing buttons appear in the grid's column headers.
filter-changing
`&` Handler for the **filterChanging** event.
filter-changed
`&` Handler for the **filterChanged** event.
filter-applied
`&` Handler for the **filterApplied** event.

WjFlexPie Class

File
`wijmo.angular.js`
Module
wijmo.angular
Derived Classes
WjSunburst

AngularJS directive for the **FlexPie** control.

items-source
= An array or **ICollectionView** object that contains data for the chart.
binding
@ The name of the property that contains item values.
binding-name
@ The name of the property that contains item names.
footer
@ The text to display in the chart footer (plain text).
footer-style
= The style to apply to the chart footer.
header
@ The text to display in the chart header (plain text).
header-style
= The style to apply to the chart header.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
inner-radius
@ The size of the hole inside the pie, measured as a fraction of the pie radius.
is-animated
@ A value indicating whether to use animation to move selected items to the selectedItemPosition.
item-formatter
= The formatter function that customizes the appearance of data points.
offset
@ The extent to which pie slices are pulled out from the center, as a fraction of the pie radius.
palette
= An array that contains the default colors used for displaying pie slices.
plot-margin
= The number of pixels of space to leave between the edges of the control and the plot area, or CSS-style margins.
reversed
@ A value indicating whether to draw pie slices in a counter-clockwise direction.
start-angle
@ The starting angle for pie slices, measured clockwise from the 9 o'clock position.
selected-item-offset
@ The extent to which the selected pie slice is pulled out from the center, as a fraction of the pie radius.
selected-item-position

@ The **Position** value indicating where to display the selected slice.

selection-mode

@ The **SelectionMode** value indicating whether or what is selected when the user clicks a series.

tooltip-content

@ The value to display in the **ChartTooltip** content property.

got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

rendering

& The **rendering** event handler.

rendered

& The **rendered** event handler.

The wj-flex-pie directive may contain the following child directives: **WjFlexChartLegend** and **WjFlexPieDataLabel**.

WjFlexPieDataLabel Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexPie PieDataLabel** object.

The **wj-flex-pie-data-label** directive must be contained in a **WjFlexPie** directive. It supports the following attributes:

content
= A string or function that gets or sets the content of the data labels.
border
@ Gets or sets a value indicating whether the data labels have borders.
position
@ The **PieLabelPosition** value indicating the position of the data labels.

WjFlexRadar Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexRadar** control.

Use the **wj-flex-radar** directive to add radar charts to your AngularJS applications. Note that directive and parameter names must be formatted using lower-case letters with dashes instead of camel case.

The wj-flex-radar directive supports the following attributes:

binding
@ The name of the property that contains Y values for the chart. You can override this at the series level.
binding-x
@ The name of the property that contains X values for the chart. You can override this at the series level.
chart-type
@ The default chart type to use in rendering series objects. You can override this at the series level. See **RadarChartType**.
control
= A reference to the **FlexRadar** control that this directive creates.
footer
@ The text to display in the chart footer (plain text).
footer-style
= The style to apply to the chart footer.
header
@ The text to display in the chart header (plain text).
header-style
= The style to apply to the chart header.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
interpolate-nulls
@ The value indicating whether to interpolate or leave gaps when there are null values in the data.
item-formatter
= The formatter function that customizes the appearance of data points.
items-source
= An array or **ICollectionView** object that contains the data used to create the chart.
legend-toggle
@ The value indicating whether clicking legend items toggles series visibility.
options
= Chart **options** that only apply to certain chart types.
palette
= An array that contains the default colors used for displaying each series.
plot-margin
= The number of pixels of space to leave between the edges of the control and the plot area, or CSS-style margins.
stacking

@ The **Stacking** value indicating whether or how series objects are stacked or plotted independently.

reversed

@ The **reversed** value indicating whether angles are reversed (counter-clockwise).

startAngle

@ The **startAngle** value indicating the starting angle for the radar in degrees.

totalAngle

@ The **totalAngle** value indicating the total angle for the radar in degrees.

symbol-size

@ The size of the symbols used to render data points in Scatter, LineSymbols, and SplineSymbols charts, in pixels. You can override this at the series level.

tooltip-content

@ The value to display in the **ChartTooltip** content property.

rendering

& The **rendering** event handler.

rendered

& The **rendered** event handler.

series-visibility-changed

& The **seriesVisibilityChanged** event handler.

The wj-flex-radar directive may contain the following child directives: **WjFlexChartAxis**, **WjFlexRadarSeries**, **WjFlexChartLegend** and **WjFlexChartDataLabel**.

WjFlexRadarAxis Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FlexRadar FlexRadarAxis Axis** object.

The **wj-flex-radar-axis** directive must be contained in a **WjFlexRadar** directive. It supports the following attributes:

wj-property
@ Defines the **FlexChart** property name, axis-x or axis-y, to initialize with the directive.
axis-line
@ The value indicating whether the axis line is visible.
binding
@ Gets or sets the comma-separated property names for the **itemsSource** property to use in axis labels. The first name specifies the value on the axis, the second represents the corresponding axis label. The default value is 'value,text'.
format
@ The format string used for the axis labels (see **Globalize**).
item-formatter
= The formatter function that customizes the appearance of axis labels.
items-source
= The items source for the axis labels.
labels
@ The value indicating whether the axis labels are visible.
label-angle
@ The rotation angle of axis labels in degrees.
label-align
@ The alignment of axis labels.
label-padding
@ The padding of axis labels.
major-grid
@ The value indicating whether the axis includes grid lines.
major-tick-marks
@ Defines the appearance of tick marks on the axis (see **TickMark**).
major-unit
@ The number of units between axis labels.
max
@ The minimum value shown on the axis.
min
@ The maximum value shown on the axis.
minor-grid
@ The value indicating whether the axis includes minor grid lines.
minor-tick-marks
@ Defines the appearance of minor tick marks on the axis (see **TickMark**).
minor-unit
@ The number of units between minor axis ticks.

origin

@ The axis origin.

overlappingLabels

@ The **OverlappingLabels** value indicating how to handle the overlapping axis labels.

position

@ The **Position** value indicating the position of the axis.

title

@ The title text shown next to the axis.

WjFlexRadarSeries Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **FinancialChart FinancialSeries** object.

The **wj-financial-chart-series** directive must be contained in a **WjFinancialChart** directive. It supports the following attributes:

axis-x
@ X-axis for the series.
axis-y
@ Y-axis for the series.
binding
@ The name of the property that contains Y values for the series. This value overrides any binding set for the chart.
binding-x
@ The name of the property that contains X values for the series. This value overrides any binding set for the chart.
chart-type
@ The chart type to use in rendering objects for this series objects. This value overrides the default chart type set on the chart. See **FinancialChartType**.
css-class
@ The CSS class to use for the series.
items-source
= An array or **ICollectionView** object that contains data for this series.
name
@ The name of the series to show in the legend.
style
= The series style. Use ng-attr-style to specify the series style object as an object. See the section on ngAttr attribute bindings in AngularJS Creating Custom Directives (https://docs.angularjs.org/guide/directive) and the FlexChart 101 Styling Series (http://demos.wijmo.com/5/Angular/FlexChartIntro/FlexChartIntro/#Styling) sample for more information.
altStyle
= The series alternative style.
symbol-marker
@ The shape of marker to use for the series. This value overrides the default marker set on the chart. See **Marker**.
symbol-size
@ The size of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts, in pixels. This value overrides any setting at the chart level.
symbol-style
= The style of the symbols used to render data points in this series for Scatter, LineSymbols, and SplineSymbols charts. This value overrides any setting at the chart level.
visibility
= The **SeriesVisibility** value indicating whether and where to display the series.

In most cases, the **wj-financial-chart-series** specifies the **name** and **binding** properties only. The remaining values are inherited from the parent **wj-financial-chart** directive.

WjFlexSheet Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjFlexGrid

AngularJS directive for the **FlexSheet** control.

Use the **wj-flex-sheet** directive to add **FlexSheet** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a FlexSheet control with one bound and two unbound sheets:</p>
<wj-flex-sheet>
   <wj-sheet name="Country" items-source="ctx.data"></wj-sheet>
   <wj-sheet name="Report" row-count="25" column-count="13"></wj-sheet>
   <wj-sheet name="Formulas" row-count="310" column-count="10"></wj-sheet>
</wj-flex-sheet>
```

The **wj-flex-sheet** directive extends **WjFlexGrid** with the following attributes:

control
`=` A reference to the **FlexSheet** control created by this directive.
is-tab-holder-visible
`@` A value indicating whether the sheet tabs holder is visible.
selected-sheet-index
`=` Gets or sets the index of the current sheet in the **FlexSheet**.
dragging-row-column
`&` Handler for the **draggingRowColumn** event.
dropping-row-column
`&` Handler for the **droppingRowColumn** event.
selected-sheet-changed
`&` Handler for the **selectedSheetChanged** event.

The **wj-flex-sheet** directive may contain **WjSheet** child directives.

WjGroupPanel Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **GroupPanel** control.

The **wj-group-panel** directive connects to the **FlexGrid** control via the **grid** property. For example:

```
<p>Here is a FlexGrid control with GroupPanel:</p>

<wj-group-panel grid="flex" placeholder="Drag columns here to create groups."></wj-group-panel>

<wj-flex-grid control="flex" items-source="data">
  <wj-flex-grid-column
    header="Country"
    binding="country">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Sales"
    binding="sales">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Expenses"
    binding="expenses">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    header="Downloads"
    binding="downloads">
  </wj-flex-grid-column>
</wj-flex-grid>
```

The **wj-group-panel** directive supports the following attributes:

grid
@ The **FlexGrid** that is connected to this **GroupPanel**.
hide-grouped-columns
@ A value indicating whether the panel hides grouped columns in the owner grid.
max-groups
@ The maximum number of groups allowed.
placeholder
@ A string to display in the control when it contains no groups.
got-focus
& Handler for the **gotFocus** event.
lost-focus
& Handler for the **lostFocus** event.

WjInputColor Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **InputColor** control.

Use the **wj-input-color** directive to add **InputColor** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an InputColor control:</p>
<wj-input-color
  value="theColor"
  show-alpha-channel="false">
</wj-input-color>
```

The **wj-input-color** directive supports the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the InputColor control created by this directive.
is-dropped-down
@ A value indicating whether the drop-down is currently visible.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
show-alpha-channel
@ A value indicating whether the drop-down displays the alpha channel (transparency) editor.
placeholder
@ The string to show as a hint when the control is empty.
is-required
@ A value indicating whether to prevent null values.
show-drop-down-button
@ A value indicating whether the control displays a drop-down button.
text
= The text to show in the control.
value
= The color being edited.
got-focus
& The **gotFocus** event handler.

lost-focus
& The **lostFocus** event handler.
is-dropped-down-changing
& The **isDroppedDownChanging** event handler.
is-dropped-down-changed
& The **isDroppedDownChanged** event handler.
text-changed
& The **textChanged** event handler.
value-changed
& The **valueChanged** event handler.

WjInputDate Class

File
`wijmo.angular.js`
Module
wijmo.angular
Derived Classes
WjInputDateTime

AngularJS directive for the **InputDate** control.

Use the **wj-input-date** directive to add **InputDate** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an InputDate control:</p>
<wj-input-date
  value="theDate"
  format="M/d/yyyy">
</wj-input-date>
```

The example below shows a **Date** value (that includes date and time information) using an **InputDate** and an **InputTime** control. Notice how both controls are bound to the same controller variable, and each edits the appropriate information (either date or time). The example also shows a **Calendar** control that can be used to select the date with a single click.

Example

Show me (http://jsfiddle.net/Wijmo5/46PhD)

The **wj-input-date** directive supports the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **InputDate** control created by this directive.
format
@ The format used to display the date being edited (see **Globalize**).
mask
@ The mask used to validate the input as the user types (see **InputMask**).
is-dropped-down
@ A value indicating whether the drop-down is currently visible.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized

= A value indicating whether the binding has finished initializing the control with attribute values.

max

@ The latest valid date (a string in the format "yyyy-MM-dd").

min

@ The earliest valid date (a string in the format "yyyy-MM-dd").

placeholder

@ The string to show as a hint when the control is empty.

is-required

@ A value indicating whether to prevent null values.

show-drop-down-button

@ A value indicating whether the control displays a drop-down button.

text

= The text to show in the control.

value

= The date being edited.

got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

is-dropped-down-changing

& The **isDroppedDownChanging** event handler.

is-dropped-down-changed

& The **isDroppedDownChanged** event handler.

text-changed

& The **textChanged** event handler.

value-changed

& The **valueChanged** event handler.

If provided, the **min** and **max** attributes are strings in the format "yyyy-MM-dd". Technically, you can use any full date as defined in the W3C **[RFC 3339]**, which is also the format used with regular HTML5 input elements.

WjInputDateTime Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjInputDate

AngularJS directive for the **InputDateTime** control.

Use the **wj-input-date-time** directive to add **InputDateTime** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an InputDateTime control:</p>
<wj-input-date-time
  value="theDate"
  format="M/d/yyyy">
</wj-input-date-time>
```

The **wj-input-date-time** directive supports the following attributes:

ng-model
`@` Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
`@` Specifies a control property that is bound to a scope using the **ng-model** directive.
control
`=` A reference to the **InputDate** control created by this directive.
format
`@` The format used to display the date being edited (see **Globalize**).
mask
`@` The mask used to validate the input as the user types (see **InputMask**).
is-dropped-down
`@` A value indicating whether the drop-down is currently visible.
initialized
`&` This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
`=` A value indicating whether the binding has finished initializing the control with attribute values.
max
`@` The latest valid date (a string in the format "yyyy-MM-dd").
min
`@` The earliest valid date (a string in the format "yyyy-MM-dd").
placeholder
`@` The string to show as a hint when the control is empty.
is-required
`@` A value indicating whether to prevent null values.

show-drop-down-button
@ A value indicating whether the control displays a drop-down button.
text
= The text to show in the control.
timeMax
@ The earliest valid time (a string in the format "hh:mm").
timeMin
@ The latest valid time (a string in the format "hh:mm").
timeStep
@ The number of minutes between entries in the drop-down list.
timeFormat
@ The format sting used to show values in the time drop-down list.
value
= The date being edited.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.
is-dropped-down-changing
& The **isDroppedDownChanging** event handler.
is-dropped-down-changed
& The **isDroppedDownChanged** event handler.
text-changed
& The **textChanged** event handler.
value-changed
& The **valueChanged** event handler.

WjInputMask Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **InputMask** control.

Use the **wj-input-mask** directive to add **InputMask** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an InputMask control:</p>
<wj-input-mask
  mask="99/99/99"
  mask-placeholder="*">
</wj-input-mask>
```

The **wj-input-mask** directive supports the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **InputNumber** control created by this directive.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
mask
@ The string mask used to format the value as the user types.
prompt-char
@ A character used to show input locations within the mask.
place-holder
@ The string to show as a hint when the control is empty.
value
= The string being edited.
raw-value
= The string being edited, excluding literal and prompt characters.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.
value-changed
& The **valueChanged** event handler.

WjInputNumber Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **InputNumber** control.

Use the **wj-input-number** directive to add **InputNumber** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an InputNumber control:</p>
<wj-input-number
  value="theNumber"
  min="0"
  max="10"
  format="n0"
  placeholder="number between zero and ten">
</wj-input-number>
```

The example below creates several **InputNumber** controls and shows the effect of using different formats, ranges, and step values.

Example

Show me (http://jsfiddle.net/Wijmo5/u7HpD)

The **wj-input-number** directive supports the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **InputNumber** control created by this directive.
format
@ The format used to display the number (see **Globalize**).
input-type
@ The "type" attribute of the HTML input element hosted by the control.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
max
@ The largest valid number.

min
@ The smallest valid number.
place-holder
@ The string to show as a hint when the control is empty.
is-required
@ A value indicating whether to prevent null values.
show-spinner
@ A value indicating whether to display spinner buttons to change the value by **step** units.
step
@ The amount to add or subtract to the value when the user clicks the spinner buttons.
text
= The text to show in the control.
value
= The number being edited.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.
text-changed
& The **textChanged** event handler.
value-changed
& The **valueChanged** event handler.

WjInputTime Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjComboBox

AngularJS directive for the **InputTime** control.

Use the **wj-input-time** directive to add **InputTime** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is an InputTime control:</p>
<wj-input-time
  value="theDate"
  format="h:mm tt"
  min="09:00" max="17:00"
  step="15">
</wj-input-time>
```

Example

Show me (http://jsfiddle.net/Wijmo5/46PhD)

This example edits a **Date** value (that includes date and time information) using an **InputDate** and an InputTime control. Notice how both controls are bound to the same controller variable, and each edits the appropriate information (either date or time). The example also shows a **Calendar** control that can be used to select the date with a single click.

The **wj-input-time** directive extends **WjComboBox** with the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **InputDate** control created by this directive.
format
@ The format used to display the selected time.
mask
@ A mask used to validate the input as the user types (see **InputMask**).
max
@ The earliest valid time (a string in the format "hh:mm").
min
@ The latest valid time (a string in the format "hh:mm").

step
@ The number of minutes between entries in the drop-down list.
value
= The time being edited (as a Date object).
value-changed
& The@see: valueChanged event handler.

If provided, the **min** and **max** attributes are strings in the format "hh:mm". Technically, you can use any full date as defined in the W3C **[RFC 3339]**, which is also the format used with regular HTML5 input elements.

WjItemTemplate Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for **ListBox** and **Menu** item templates.

The **wj-item-template** directive must be contained in a **WjListBox** or **WjMenu** directives.

The **wj-item-template** directive defines a template for items of **ListBox** and data-bound **Menu** controls. The template may contain an arbitrary HTML fragment with AngularJS bindings and directives. In addition to any properties available in a controller, the local **$item**, **$itemIndex** and **$control** template variables can be used in AngularJS bindings that refer to the data item, its index, and the owner control.

Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a ListBox control with an item template:</p>
<wj-list-box items-source="musicians">
    <wj-item-template>
        {{$itemIndex}}. <b>{{$item.name}}</b>
        <br />
        <img ng-src="{{$item.photo}}"/>
    </wj-item-template>
</wj-list-box>
```

WjLinearGauge Class

File
`wijmo.angular.js`
Module
wijmo.angular
Derived Classes
WjBulletGraph

AngularJS directive for the **LinearGauge** control.

Use the **wj-linear-gauge** directive to add linear gauges to your AngularJS applications. Note that directive and parameter names must be formatted in lower-case with dashes instead of camel-case. For example:

```
<wj-linear-gauge
  value="ctx.gauge.value"
  show-text="Value"
  is-read-only="false">
  <wj-range
    wj-property="pointer"
    thickness="0.2">
    <wj-range
      min="0"
      max="33"
      color="green">
    </wj-range>
    <wj-range
      min="33"
      max="66"
      color="yellow">
    </wj-range>
    <wj-range
      min="66"
      max="100"
      color="red">
    </wj-range>
  </wj-range>
</wj-linear-gauge>
```

The **wj-linear-gauge** directive supports the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **LinearGauge** control created by this directive.
direction
@ The **GaugeDirection** value in which the gauge fills as the value grows.
format

@ The format string used for displaying the gauge values as text.

has-shadow

@ A value indicating whether the gauge displays a shadow effect.

initialized

& This event occurs after the binding has finished initializing the control with attribute values.

is-initialized

= A value indicating whether the binding has finished initializing the control with attribute values.

is-animated

@ A value indicating whether the gauge animates value changes.

is-read-only

@ A value indicating whether users are prevented from editing the value.

min

@ The minimum value that the gauge can display.

max

@ The maximum value that the gauge can display.

show-text

@ The **ShowText** value indicating which values display as text within the gauge.

step

@ The amount to add or subtract to the value property when the user presses the arrow keys.

thickness

@ The thickness of the gauge, on a scale of zero to one.

value

= The value displayed on the gauge.

got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

The **wj-linear-gauge** directive may contain one or more **WjRange** directives.

Example

Show me (http://jsfiddle.net/Wijmo5/t842jozb)

WjListBox Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **ListBox** control.

Use the **wj-list-box** directive to add **ListBox** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
b>Here is a ListBox control:</p>
<wj-list-box
  selected-item="theCountry"
  items-source="countries"
  placeholder="country">
</wj-list-box>
```

The example below creates a **ListBox** control and binds it to a 'countries' array exposed by the controller. The value selected is bound to the 'theCountry' controller property using the **selected-item** attribute.

Example

Show me (http://jsfiddle.net/Wijmo5/37GHw)

The **wj-list-box** directive supports the following attributes:

ng-model
@ Binds the control's **selectedValue** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the **ListBox** control created by this directive.
display-member-path
@ The property to use as the visual representation of the items.
is-content-html
@ A value indicating whether items contain plain text or HTML.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
item-formatter
= A function used to customize the values to show in the list.
items-source
= An array or **ICollectionView** that contains the list items.
max-height

@ The maximum height of the list.

selected-index

= The index of the currently selected item.

selected-item

= The item that is currently selected.

selected-value

= The value of the **selected-item** obtained using the **selected-value-path**.

selected-value-path

@ The property used to get the **selected-value** from the **selected-item**.

got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

items-changed

& The **itemsChanged** event handler.

selected-index-changed

& The **selectedIndexChanged** event handler.

The **wj-list-box** directive may contain **WjItemTemplate** child directive.

WjMenu Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjComboBox

AngularJS directive for the **Menu** control.

Use the **wj-menu** directive to add drop-down menus to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a Menu control used as a value picker:</p>
<wj-menu header="Tax" value="tax">
  <wj-menu-item value="0">Exempt</wj-menu-item>
  <wj-menu-item value=".05">5%</wj-menu-item>
  <wj-menu-item value=".1">10%</wj-menu-item>
  <wj-menu-item value=".15">15%</wj-menu-item>
</wj-menu>
```

Example

Show me (http://jsfiddle.net/Wijmo5/Wc5Mq)

This example creates three **Menu** controls. The first is used as a value picker, the second uses a list of commands with parameters, and the third is a group of three menus handled by an **itemClicked** function in the controller.

The **wj-menu** directive extends **WjComboBox** with the following attributes:

command-path
@ The command to be executed when the item is clicked.
command-parameter-path
@ The name of the property that contains command parameters.
header
@ The text shown on the control.
is-button
@ Whether the menu should react to clicks on its header area.
value
@ The value of the selected **wj-menu-item** value property.
item-clicked
& The **itemClicked** event handler.
got-focus
& The **gotFocus** event handler.
lost-focus
& The **lostFocus** event handler.

The **wj-menu** directive may contain the following child directives: **WjMenuItem**, **WjMenuSeparator** and **WjItemTemplate**(in case of data-bound Menu control).

WjMenuItem Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for menu items.

The **wj-menu-item** directive must be contained in a **WjMenu** directive. It supports the following attributes:

cmd
= The function to execute in the controller when the item is clicked.
cmd-param
= The parameter passed to the **cmd** function when the item is clicked.
value
= The value to select when the item is clicked (use either this or **cmd**).

The content displayed by the item may contain an arbitrary HTML fragment with AngularJS bindings and directives. You can also use **ng-repeat** and **ng-if** directives to populate the items in the Menu control. In addition to any properties available in a controller, the local **$item**, **$itemIndex** and **$control** template variables can be used in AngularJS bindings that refer to the data item, its index, and the owner control.

WjMenuSeparator Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for menu separators.

The **wj-menu-item-separator** directive must be contained in a **WjMenu** directive. It adds a non-selectable separator to the menu, and has no attributes.

WjMultiAutoComplete Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjAutoComplete

AngularJS directive for the **MultiAutoComplete** control.

Use the **wj-multi-auto-complete** directive to add **MultiAutoComplete** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a InputTags bound to a collection of objects:</p>
<wj-multi-auto-complete
    max-selected-items="8"
    items-source="ctx.items"
    selected-Member-path="selected">
</wj-multi-auto-complete>
```

The **wj-multi-auto-complete** directive extends **WjAutoComplete** with the following attributes:

max-selected-items
`@` The maximum number of items that can be selected.
selected-member-path
`@` The name of the property used to control which item will be selected.
selected-items
`@` An array containing the items that are currently selected.
selected-items-changed
`&` The **selectedItemsChanged** event handler.

WjMultiRow Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjFlexGrid

AngularJS directive for the **MultiRow** control.

Use the **wj-multi-row** directive to add **MultiRow** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case.

The **wj-multi-row** directive extends **WjFlexGrid** with the following attributes:

control
= A reference to the **MultiRow** control created by this directive.
layout-definition
@ A value defines the layout of the rows used to display each data item.
collapsed-headers
@ Gets or sets a value that determines whether column headers should be collapsed and displayed as a single row displaying the group headers.
center-headers-vertically
@ Gets or sets a value that determines whether the content of cells that span multiple rows should be vertically centered.
show-header-collapse-button
@ Gets or sets a value that determines whether the grid should display a button in the column header panel to allow users to collapse and expand the column headers.

WjMultiSelect Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjComboBox

AngularJS directive for the **MultiSelect** control.

Use the **wj-multi-select** directive to add **MultiSelect** controls to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a MultiSelect bound to a collection of objects:</p>
<wj-multi-select
    placeholder="Select Countries"
    items-source="ctx.items"
    header-format="{count} countries selected"
    display-Member-path="country"
    checked-Member-path="selected">
</wj-multi-select>
```

The **wj-multi-select** directive extends **WjComboBox** with the following attributes:

checked-member-path
`@` The name of the property used to control the checkboxes placed next to each item.
header-format
`@` The format string used to create the header content when the control has more than **maxHeaderItems** items checked.
header-formatter
`=` A function that gets the HTML in the control header.
max-header-items
`@` The maximum number of items to display on the control header.
checked-items-changed
`&` The **checkedItemsChanged** event handler.

WjPdfViewer Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **PdfViewer** control.

Use the **wj-pdf-viewer** directive to add pdf viewer to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<wj-pdf-viewer  paginated="false"
  service-url="ctx.serviceUrl"
  file-path="ctx.path"
  report-name="ctx.reportName">
</wj-pdf-viewer;
```

The **wj-pdf-viewer** directive supports the following attributes:

service-url
@ A value indicating the address of C1 Web API service.
file-path
@ A value indicating full path to the document on the server.
control
= A reference to the **PdfViewer** control created by this directive.
full-screen
@ A value indicating whether viewer is under full screen mode.
zoom-factor
@ A value indicating the current zoom factor to show the document pages
mouse-mode
@ A value indicating the mouse behavior.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
view-mode
@ A value indicating how to show the document pages.
page-index-changed
& The **pageIndexChanged** event handler.
view-mode-changed
& The **viewModeChanged** event handler.
mouse-mode-changed
& The **mouseModeChanged** event handler.
full-screen-changed
& The **fullScreenChanged** event handler.
zoom-factor-changed
& The **zoomFactorChanged** event handler.

query-loading-data

& The **queryLoadingData** event handler.

WjPivotChart Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **PivotChart** control.

Use the **wj-pivot-chart** and **wj-pivot-panel** directives to add pivot charts to your AngularJS applications.

Directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<wj-pivot-panel
    control="thePanel"
    items-source="rawData">
</wj-pivot-panel>
<wj-pivot-chart
    items-source="thePanel"
    chart-type="Bar"
    max-series="10"
    max-points="100">
</wj-pivot-chart>
```

The **wj-pivot-chart** directive supports the following attributes:

items-source
Gets or sets the **PivotPanel** that defines the view displayed by this **PivotChart**.
chart-type
Gets or sets a **PivotChartType** value that defines the type of chart to display.
show-hierarchical-axes
Gets or sets whether the chart should group axis annotations for grouped data.
stacking
Gets or sets a **Stacking** value that determines whether and how the series objects are stacked.
show-totals
Gets or sets a whether the chart should include only totals.
max-series
Gets or sets the maximum number of data series to be shown in the chart.
max-points
Gets or sets the maximum number of points to be shown in each series.

WjPivotGrid Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjFlexGrid

AngularJS directive for the **PivotGrid** control.

Use the **wj-pivot-grid** and **wj-pivot-panel** directives to add pivot tables to your AngularJS applications.

Directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<wj-pivot-panel
    control="thePanel"
    items-source="rawData">
</wj-pivot-panel>
<wj-pivot-grid
    items-source="thePanel"
    show-detail-on-double-click="false"
    custom-context-menu="true">
</wj-pivot-grid>
```

The **wj-pivot-grid** directive extends the **wj-flex-grid** directive and adds support for the following attributes:

items-source
Gets or sets the **PivotPanel** that defines the view displayed by this **PivotGrid**.
show-detail-on-double-click
Gets or sets whether the grid should show a popup containing the detail records when the user double-clicks a cell.
custom-context-menu
Gets or sets whether the grid should provide a custom context menu with commands for changing field settings and showing detail records.
collapsible-subtotals
Gets or sets whether the grid should allow users to collapse and expand subtotal groups of rows and columns.
center-headers-vertically
Gets or sets whether the content of header cells should be vertically centered.

WjPivotPanel Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **PivotPanel** control.

Use the **wj-pivot-panel** directive as a data source for **wj-pivot-grid** and **wj-pivot-chart** directives to add pivot tables and charts to your AngularJS applications.

Directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<wj-pivot-panel
    control="thePanel"
    items-source="rawData">
</wj-pivot-panel>
<wj-pivot-grid
    items-source="thePanel"
    show-detail-on-double-click="false"
    custom-context-menu="true">
</wj-pivot-grid>
```

The **wj-pivot-panel** directive supports the following attributes:

items-source
Gets or sets the raw data used to generate pivot views.
auto-generate-fields
Gets or sets whether the panel should populate its fields collection automatically based on the **itemsSource**.
view-definition
Gets or sets the current pivot view definition as a JSON string.
engine
Gets a reference to the **PivotEngine** that summarizes the data.

WjPopup Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **Popup** control.

Use the **wj-popup** directive to add **Popup** controls to your AngularJS applications.

The popup content may be specified inside the **wj-popup** tag, and can contain an arbitrary HTML fragment with AngularJS bindings and directives.

Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<p>Here is a Popup control triggered by a button:</p>
<button id="btn2" type="button">
    Click to show Popup
</button>
<wj-popup owner="#btn2" show-trigger="Click" hide-trigger="Blur">
    <h3>
        Salutation
    </h3>
    <div class="popover-content">
        Hello {{firstName}} {{lastName}}
    </div>
</wj-popup>
```

The **wj-popup** directive supports the following attributes:

control
`=` A reference to the Popup control created by this directive.
fade-in
`@` A boolean value that determines whether popups should be shown using a fade-in animation.
fade-out
`@` A boolean value that determines whether popups should be hidden using a fade-out animation.
hide-trigger
`@` A **PopupTrigger** value defining the action that hides the **Popup**.
initialized
`&` This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
`=` A value indicating whether the binding has finished initializing the control with attribute values.
owner
`@` A CSS selector referencing an element that controls the popup visibility.
show-trigger
`@` A **PopupTrigger** value defining the action that shows the **Popup**.
modal
`@` A boolean value that determines whether the **Popup** should be displayed as a modal dialog.
got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

showing

& The **showing** event handler.

shown

& The **shown** event handler.

hiding

& The **hiding** event handler.

hidden

& The **hidden** event handler.

WjRadialGauge Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **RadialGauge** control.

Use the **wj-radial-gauge** directive to add radial gauges to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
Here is a <b>RadialGauge</b> control:</p>
<wj-radial-gauge
  style="height:300px"
  value="count"
  min="0" max="10"
  is-read-only="false">
</wj-radial-gauge>
```

The **wj-radial-gauge** directive supports the following attributes:

ng-model
@ Binds the control's **value** property using the ng-model Angular directive. Binding the property using the ng-model directive provides standard benefits like validation, adding the control's state to the form instance, and so on. To redefine properties on a control that is bound by the ng-model directive, use the wj-model-property attribute.
wj-model-property
@ Specifies a control property that is bound to a scope using the **ng-model** directive.
control
= A reference to the RadialGauge control created by this directive.
auto-scale
@ A value indicating whether the gauge scales the display to fill the host element.
format
@ The format string used for displaying gauge values as text.
has-shadow
@ A value indicating whether the gauge displays a shadow effect.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
is-animated
@ A value indicating whether the gauge animates value changes.
is-read-only
@ A value indicating whether users are prevented from editing the value.
min
@ The minimum value that the gauge can display.
max
@ The maximum value that the gauge can display.
show-text

@ A **ShowText** value indicating which values display as text within the gauge.

step

@ The amount to add or subtract to the value property when the user presses the arrow keys.

start-angle

@ The starting angle for the gauge, in degreees, measured clockwise from the 9 o'clock position.

sweep-angle

@ The sweeping angle for the gauge in degrees (may be positive or negative).

thickness

@ The thickness of the gauge, on a scale of zero to one.

value

= The value displayed on the gauge.

got-focus

& The **gotFocus** event handler.

lost-focus

& The **lostFocus** event handler.

The **wj-radial-gauge** directive may contain one or more **WjRange** directives.

Example

Show me (http://jsfiddle.net/Wijmo5/7ec2144u)

WjRange Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **Range** object.

The **wj-range** directive must be contained in a **WjLinearGauge**, **WjRadialGauge** or **WjBulletGraph** directive. It adds the Range object to the 'ranges' array property of the parent directive. You may also initialize other Range type properties of the parent directive by specifying the property name with the wj-property attribute.

For example:

```
<wj-radial-gauge
    min="0"
    max="200"
    step="20"
    value="theValue"
    is-read-only="false">
    <wj-range
      min="0"
      max="100"
      color="red">
    </wj-range>
    <wj-range
      min="100"
      max="200"
      color="green">
    </wj-range>
    <wj-range
      wj-property="pointer"
      color="blue">
    </wj-range>
</wj-radial-gauge>
```

The **wj-range** directive supports the following attributes:

min
@ The minimum value in the range.
max
@ The maximum value in the range.
color
@ The color used to display the range.
thickness
@ The thickness of the range, on a scale of zero to one.
name
@ The name of the range.
wj-property
@ The name of the property to initialize with this directive.

WjReportViewer Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **ReportViewer** control.

Use the **wj-report-viewer** directive to add report viewer to your AngularJS applications. Note that directive and parameter names must be formatted in lower-case with dashes instead of camel-case. For example:

```
<wj-report-viewer paginated="true"
  service-url="ctx.serviceUrl"
  file-path="ctx.path"
  report-name="ctx.reportName">
</wj-report-viewer;
```

The **wj-report-viewer** directive supports the following attributes:

service-url
@ A value indicating the address of C1 Web API service.
file-path
@ A value indicating full path to the document on the server.
report-name
@ For FlexReport, sets it with the report name defined in the FlexReport definition file. For SSRS report, please leave it as empty string.
control
= A reference to the **ReportViewer** control created by this directive.
full-screen
@ A value indicating whether viewer is under full screen mode.
zoom-factor
@ A value indicating the current zoom factor to show the document pages
mouse-mode
@ A value indicating the mouse behavior.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
view-mode
@ A value indicating how to show the document pages.
paginated
@ A value indicating whether the content should be represented as set of fixed sized pages.
page-index-changed
& The **pageIndexChanged** event handler.
view-mode-changed
& The **viewModeChanged** event handler.
mouse-mode-changed

& The **mouseModeChanged** event handler.

full-screen-changed

& The **fullScreenChanged** event handler.

zoom-factor-changed

& The **zoomFactorChanged** event handler.

query-loading-data

& The **queryLoadingData** event handler.

WjSheet Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **Sheet** object.

The **wj-sheet** directive must be contained in a **WjFlexSheet** directive. It supports the following attributes:

name
`@` The name of the sheet.
row-count
`@` The initial number of rows in the unbound sheet. Changes done to this attribute have no effect after the **Sheet** was initialized by AngularJS.
column-count
`@` The initial number of columns in the unbound sheet. Changes done to this attribute have no effect after the **Sheet** was initialized by AngularJS.
items-source
`=` The data source for the data bound sheet. Changes done to this attribute have no effect after the **Sheet** was initialized by AngularJS.
visible
`@` A value indicating whether the sheet is visible.
name-changed
`&` Handler for the **nameChanged** event.

WjSunburst Class

File
`wijmo.angular.js`
Module
wijmo.angular
Base Class
WjFlexPie

AngularJS directive for the **Sunburst** control.

child-items-path
= An array or string object used to generate child items in hierarchical data.

WjTooltip Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **Tooltip** class.

Use the **wj-tooltip** directive to add tooltips to elements on the page. The wj-tooltip directive supports HTML content, smart positioning, and touch.

The wj-tooltip directive is specified as a parameter added to the element that the tooltip applies to. The parameter value is the tooltip text or the id of an element that contains the text. For example:

```
<p wj-tooltip="#fineprint" >
    Regular paragraph content...</p>
...
<div id="fineprint" style="display:none">
  <h3>Important Note</h3>
  <p>
    Data for the current quarter is estimated
    by pro-rating etc.</p>
</div>
```

WjTreeMap Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **TreeMap** control.

items-source
= An array or **ICollectionView** object that contains data for the chart.
binding
@ The name of the property that contains item values.
binding-name
@ The name of the property that contains item names. It should be an array or a string.
footer
@ The text to display in the chart footer (plain text).
footer-style
= The style to apply to the chart footer.
header
@ The text to display in the chart header (plain text).
header-style
= The style to apply to the chart header.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
type
@ The type of the TreeMap.
child-items-path
@ A value indicating the name of the property (or properties) used to generate child items in hierarchical data.
max-depth
= The maximum number of node levels to show in the current view.
palette
= An array that contains the default colors used for displaying TreeMap items.
plot-margin
= The number of pixels of space to leave between the edges of the control and the plot area, or CSS-style margins.
tooltip-content
@ The value to display in the **ChartTooltip** content property.
label-content
@ The value to display in the **DataLabel** content property.
rendering
& The **rendering** event handler.
rendered
& The **rendered** event handler.

The wj-tree-map directive may contain the following child directives: **WjFlexChartLegend** and **WjFlexChartDataLabel**.

WjTreeView Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for the **TreeView** control.

Use the **wj-tree-view** directive to add TreeView to your AngularJS applications. Note that directive and parameter names must be formatted as lower-case with dashes instead of camel-case. For example:

```
<wj-tree-view  items-source="items"
  display-member-path="ctx.displayMemberPath"
  child-items-path="ctx.childItemsPath"
  is-animated="ctx.isAnimated">
<wj-tree-view;
```

The **wj-tree-view** directive supports the following attributes:

items-source
= An array that contains the **TreeView** items.
child-items-path
@ A value indicating the name of the property (or properties) that contains the child items for each node.
control
= A reference to the **TreeView** control created by this directive.
display-member-path
@ A value indicating the name of the property (or properties) to use as the visual representation of the nodes.
image-member-path
@ A value indicating the name of the property (or properties) to use as a source of images for the nodes.
is-content-html
@ A value indicating whether whether items are bound to plain text or HTML.
initialized
& This event occurs after the binding has finished initializing the control with attribute values.
is-initialized
= A value indicating whether the binding has finished initializing the control with attribute values.
show-checkboxes
@ A value determines whether the **TreeView** should add checkboxes to nodes and manage their state.
auto-collapse
@ A value determines if sibling nodes should be collapsed when a node is expanded.
is-animated
@ A value indicating whether to use animations when expanding or collapsing nodes.
is-readOnly
@ A value determines whether users can edit the text in the nodes.
allow-dragging
@ A value determines whether users can drag and drop nodes within the **TreeView**.
expand-on-click
@ A value determines whether to expand collapsed nodes when the user clicks the node header.

selected-item
@ A value indicating the data item that is currently selected.
selected-node
@ A value indicating **TreeNode** that is currently selected.
checked-items
@ An array containing the items that are currently checked.
lazy-load-function
= A function that loads child nodes on demand.
items-source-changed
& The **itemsSourceChanged** event handler.
loading-items
& The **loadingItems** event handler.
loaded-items
& The **loadedItems** event handler.
item-clicked
& The **itemClicked** event handler.
selected-item-changed
& The **selectedItemChanged** event handler.
checked-items-Changed
& The **checkedItemsChanged** event handler.
is-collapsed-changing
& The **isCollapsedChanging** event handler.
is-collapsed-changed
& The **isCollapsedChanged** event handler.
is-checked-changing
& The **isCheckedChanging** event handler.
is-checked-changed
& The **isCheckedChanged** event handler.
format-item
& The **formatItem** event handler.
drag-start
& The **dragStart** event handler.
drag-over
& The **dragOver** event handler.
drop
& The **drop** event handler.
drag-end
& The **dragEnd** event handler.
node-edit-starting
& The **nodeEditStarting** event handler.
node-edit-started
& The **nodeEditStarted** event handler.
node-edit-ending
& The **nodeEditEnding** event handler.
node-edit-ended
& The **nodeEditEnded** event handler.

WjValidationError Class

File
`wijmo.angular.js`
Module
wijmo.angular

AngularJS directive for custom validations based on expressions.

The **wj-validation-error** directive supports both AngularJS and native HTML5 validation mechanisms. It accepts an arbitrary AngularJS expression that should return an error message string in case of the invalid input and an empty string if the input is valid.

For AngularJS validation it should be used together with the **ng-model** directive. In this case the **wj-validation-error** directive reports an error using a call to the **NgModelController.$setValidity** method with the **wjValidationError** error key , in the same way as it happens with AngularJS native and custom validation directives.

For HTML5 validation, the **wj-validation-error** directive sets the error state to the element using the **setCustomValidity** method from the HTML5 validation API. For example:

```
<p>HTML5 validation:</p>
<form>
    <input type="password"
        placeholder="Password"
        name="pwd" ng-model="thePwd"
        required minlength="2" />
    <input type="password"
        placeholder="Check Password"
        name="chk" ng-model="chkPwd"
        wj-validation-error="chkPwd != thePwd ? 'Passwords don\'t match' : ''" />
</form>

<p>AngularJS validation:</p>
<form name="ngForm" novalidate>
    <input type="password"
        placeholder="Password"
        name="pwd" ng-model="thePwd"
        required minlength="2" />
    <input type="password"
        placeholder="Check Password"
        name="chk" ng-model="chkPwd"
        wj-validation-error="chkPwd != thePwd" />
    <div
        ng-show="ngForm.chk.$error.wjValidationError && !ngForm.chk.$pristine">
        Sorry, the passwords don't match.
    </div>
</form>
```

CellTemplateType Enum

File
`wijmo.angular.js`
Module
wijmo.angular

Defines the type of cell to which the template applies. This value is specified in the **cell-type** attribute of the **WjFlexGridCellTemplate** directive.

Members

Name	Value	Description
Cell	0	Defines a regular (data) cell.
CellEdit	1	Defines a cell in edit mode.
ColumnHeader	2	Defines a column header cell.
RowHeader	3	Defines a row header cell.
RowHeaderEdit	4	Defines a row header cell in edit mode.
TopLeft	5	Defines a top left cell.
GroupHeader	6	Defines a group header cell in a group row.
Group	7	Defines a regular cell in a group row.
ColumnFooter	8	Defines a column footer cell.
BottomLeft	9	Defines a bottom left cell (at the intersection of the row header and column footer cells).

wijmo.knockout Module

File
`wijmo.knockout.js`
Module
wijmo.knockout

Contains KnockoutJS bindings for the Wijmo controls.

The bindings allow you to add Wijmo controls to **KnockoutJS** applications using simple markup in HTML pages.

To add a Wijmo control to a certain place in a page's markup, add the ** ** element and define a binding for the control in the **data-bind** attribute. The binding name corresponds to the control name with a wj prefix. For example, the **wjInputNumber** binding represents the Wijmo **InputNumber** control. The binding value is an object literal containing properties corresponding to the control's read-write property and event names, with their values defining the corresponding control property values and event handlers.

The following markup creates a Wijmo **InputNumber** control with the **value** property bound to the view model's **theValue** property, the **step** property set to 1 and the **inputType** property set to 'text':

```
<div data-bind="wjInputNumber: { value: theValue, step: 1, inputType: 'text' }"></div>
```

Custom elements

As an alternative to the standard Knockout binding syntax, the Wijmo for Knockout provides a possibility to declare controls in the page markup as custom elements, where the tag name corresponds to the control binding name and the attribute names correspond to the control property names. The element and parameter names must be formatted as lower-case with dashes instead of camel-case. The control in the example above can be defined as follows using the custom element syntax:

```
<wj-input-number value="theValue" step="1" input-type="'text'"></wj-input-number>
```

Note that attribute values should be defined using exactly the same JavaScript expressions syntax as you use in data-bind definitions. The Wijmo for Knockout preprocessor converts such elements to the conventional data-bind form, see the **Custom elements preprocessor** topic for more details.

Binding to control properties

Wijmo binding for KnockoutJS supports binding to any read-write properties on the control. You can assign any valid KnockoutJS expressions (e.g. constants, view model observable properties, or complex expressions) to the property.

Most of the properties provide one-way binding, which means that changes in the bound observable view model property cause changes in the control property that the observable is bound to, but not vice versa. But some properties support two-way binding, which means that changes made in the control property are propagated back to an observable bound to the control property as well. Two-way bindings are used for properties that can be changed by the control itself, by user interaction with the control, or by other occurences. For example, the InputNumber control provides two-way binding for the **value** and **text** properties, which are changed by the control while a user is typing a new value. The rest of the InputNumber properties operate in the one-way binding mode.

Binding to control events

To attach a handler to a control event, specify the event name as a property of the object literal defining the control binding, and the function to call on this event as a value of this property. Wijmo bindings follow the same rules for defining an event handler as used for the intrinsic KnockoutJS bindings like **click** and **event**. The event handler receives the following set of parameters, in the specified order:

- **data:** The current model value, the same as for native KnockoutJS bindings like **click** and **event**.
- **sender:** The sender of the event.
- **args:** The event arguments.

The following example creates an **InputNumber** control and adds an event handler for the **valueChanged** event showing a dialog with a new control value.

```
<!-- HTML -->
<div data-bind="wjInputNumber: { value: theValue, step: 1, valueChanged: valueChangedEH }"></div>
```

```
//View Model
this.valueChangedEH = function (data, sender, args) {
    alert('The new value is: ' + sender.value);
}
```

The same control defined using the custom element syntax:

```
<wj-input-number value="theValue" step="1" value-changed="valueChangedEH"></wj-input-number>
```

Binding to undefined observables

View model observable properties assigned to an *undefined* value get special treatment by Wijmo bindings during the initialization phase. For example, if you create an observable as ko.observable(undefined) or ko.observable() and bind it to a control property, Wijmo does not assign a value to the control. Instead, for properties supporting two-way bindings, this is the way to initialize the observable with the control's default value, because after initialization the control binding updates bound observables with the control values of such properties. Note that an observable with a *null* value, e.g. ko.observable(null), gets the usual treatment and assigns null to the control property that it is bound to. After the primary initialization has finished, observables with undefined values go back to getting the usual treatment from Wijmo, and assign the control property with undefined.

In the example below, the **value** property of the **InputNumber** control has its default value of 0 after initialization, and this same value is assigned to the view model **theValue** property:

```
<!-- HTML -->
<div data-bind="wjInputNumber: { value: theValue }"></div>
```

```
//View Model
this.theValue = ko.observable();
```

Defining complex and array properties

Some Wijmo controls have properties that contain an array or a complex object. For example, the **FlexChart** control exposes **axisX** and **axisY** properties that represent an **Axis** object; and the **series** property is an array of **Series** objects. Wijmo provides special bindings for such types that we add to child elements of the control element. If the control exposes multiple properties of the same complex type, then the **wjProperty** property of the complex type binding specifies which control property it defines.

The following example shows the markup used to create a **FlexChart** with **axisX** and **axisY** properties and two series objects defined:

```
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: chartProps.titleX }"></div>
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisY', title: chartProps.titleY }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Downloads', binding: 'downloads' }"></div>
</div>
```

The same control defined using the custom element syntax:

```
<wj-flex-chart items-source="data" binding-x="'country'">
    <wj-flex-chart-axis wj-property="'axisX'" title="chartProps.titleX"></wj-flex-chart-axis>
    <wj-flex-chart-axis wj-property="'axisY'" title="chartProps.titleY"></wj-flex-chart-axis>
    <wj-flex-chart-series name="'Sales'" binding"'sales'"></wj-flex-chart-series>
    <wj-flex-chart-series name="'Downloads'" binding"'downloads'"></wj-flex-chart-series>
</wj-flex-chart>
```

The **control** property

Each Wijmo control binding exposes a **control** property that references the Wijmo control instance created by the binding. This allows you to reference the control in view model code or in other bindings. For example, the following markup creates a **FlexGrid** control whose reference is stored in the **flex** observable property of a view model and is used in the button click event handler to move to the next grid record:

```
<!-- HTML -->
<div data-bind="'wjFlexGrid': { itemsSource: data, control: flex }"></div>
<button data-bind="click: moveToNext">Next</button>
```

```
//View Model
this.flex = ko.observable();
this.moveToNext = function () {
    this.flex().collectionView.moveCurrentToNext();
}
```

The **initialized** event

Each Wijmo control binding exposes an **initialized** event and a Boolean **isInitialized** property. The event occurs right after the binding creates the control and fully initializes it with the values specified in the binding attributes. For bindings containing child bindings, for example, a **wjFlexGrid** with child **wjFlexGridColumn** bindings, this also means that child bindings have fully initialized and have been applied to the control represented by the parent binding. The isInitialized property is set to true right before triggering the initialized event. You can bind a view model observable property to the binding's **isInitialized** property to access its value.

The following example adjusts FlexGridColumn formatting after the control fully initializes with its bindings, which guarantees that these formats are not overwritten with formats defined in the bindings:

```
<!-- HTML -->
<div data-bind="'wjFlexGrid': { itemsSource: dataArray, initialized: flexInitialized }">
     <div data-bind="wjFlexGridColumn: { binding: 'sales', format: 'n2' }"></div>
     <div data-bind="wjFlexGridColumn: { binding: 'downloads', format: 'n2' }"></div>
</div>


//View Model
this.flexInitialized = function (data, sender, args) {
    var columns = sender.columns;
    for (var i = 0; i < columns.length; i++) {
        if (columns[i].dataType = wijmo.DataType.Number) {
            columns[i].format = 'n0’;
        }
    }
}
```

Custom elements preprocessor

The Wijmo Knockout preprocessor uses the standard Knockout **ko.bindingProvider.instance.preprocessNode** API. This may cause problems in cases where other custom preprocessors are used on the same page, because Knockout offers a single instance property for attaching a preprocessor function, and the next registering preprocessor removes the registration of the previous one.

To honor another attached preprocessor, the Wijmo Knockout preprocessor stores the currently registered preprocessor during initialization and delegates the work to it in cases where another processing node is not recognized as a Wijmo control element, thus organizing a preprocessor stack. But if you register another preprocessor after the Wijmo for Knockout preprocessor (that is, after the <script> reference to the **wijmo.knockout.js** module is executed) then you need to ensure that the other preprocessor behaves in a similar way; otherwise, the Wijmo Knockout preprocessor is disabled.

If you prefer to disable the Wijmo Knockout preprocessor, set the **wijmo.disableKnockoutTags** property to false before the **wijmo.knockout.js** module reference and after the references to the core Wijmo modules, for example:

```
<script src="scripts/wijmo.js"></script>
<script src="scripts/wijmo.input.js"></script>
<script>
    wijmo.disableKnockoutTags = true;
</script>
<script src="scripts/wijmo.knockout.js"></script>
```

Note that in this case you can use only the conventional data-bind syntax for adding Wijmo controls to the page markup; the Wijmo custom elements are not recognized.

Classes

- wjAutoComplete
- wjBulletGraph
- wjCalendar
- wjCollectionViewNavigator
- wjCollectionViewPager
- wjColorPicker
- wjComboBox
- wjContextMenu
- wjFinancialChart
- wjFinancialChartSeries
- wjFlexChart
- wjFlexChartAnimation
- wjFlexChartAnnotation
- wjFlexChartAnnotationLayer
- wjFlexChartAtr
- wjFlexChartAxis
- wjFlexChartBollingerBands
- wjFlexChartCci
- wjFlexChartDataPoint
- wjFlexChartEnvelopes
- wjFlexChartFibonacci
- wjFlexChartFibonacciArcs
- wjFlexChartFibonacciFans
- wjFlexChartFibonacciTimeZones
- wjFlexChartGestures
- wjFlexChartLegend
- wjFlexChartLineMarker
- wjFlexChartMacd
- wjFlexChartMacdHistogram
- wjFlexChartMovingAverage
- wjFlexChartParametricFunctionSeries
- wjFlexChartPlotArea
- wjFlexChartRangeSelector
- wjFlexChartRsi
- wjFlexChartSeries
- wjFlexChartStochastic
- wjFlexChartTrendLine
- wjFlexChartWaterfall
- wjFlexChartWilliamsR
- wjFlexChartYFunctionSeries
- wjFlexGrid
- wjFlexGridColumn
- wjFlexGridFilter
- wjFlexPie
- wjFlexSheet
- wjGroupPanel
- wjInputColor
- wjInputDate
- wjInputDateTime
- wjInputMask
- wjInputNumber
- wjInputTime
- wjLinearGauge
- wjListBox
- wjMenu
- wjMenuItem
- wjMenuSeparator
- wjMultiAutoComplete
- wjMultiRow
- wjMultiSelect
- wjPivotChart
- wjPivotGrid
- wjPivotPanel
- wjPopup
- wjRadialGauge
- wjRange
- wjSheet
- wjStyle
- wjTooltip
- wjTreeView

wjAutoComplete Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjComboBox
Derived Classes
wjMultiAutoComplete

KnockoutJS binding for the **AutoComplete** control.

Use the **wjAutoComplete** binding to add **AutoComplete** controls to your KnockoutJS applications. For example:

```
<p>Here is an AutoComplete control:</p>
<div data-bind="wjAutoComplete: {
  itemsSource: countries,
  text: theCountry,
  isEditable: false,
  placeholder: 'country' }">
</div>
```

The **wjAutoComplete** binding supports all read-write properties and events of the **AutoComplete** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjBulletGraph Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjLinearGauge

KnockoutJS binding for the **BulletGraph** control.

Use the **wjBulletGraph** binding to add **BulletGraph** controls to your KnockoutJS applications. For example:

```
<p>Here is a BulletGraph control:</p>
<div data-bind="wjBulletGraph: {
        value: props.value,
        min: props.min,
        max: props.max,
        format: props.format,
        good: props.ranges.middle.max,
        bad: props.ranges.middle.min,
        target: props.ranges.target,
        showRanges: props.showRanges }"
        class="linear-gauge">
    <div data-bind="wjRange: {
            wjProperty: 'pointer',
            thickness: props.ranges.pointerThickness }">
    </div>
</div>
```

The **wjBulletGraph** binding may contain the **wjRange** child binding.

The **wjBulletGraph** binding supports all read-write properties and events of the **BulletGraph** control. The **value** property provides two-way binding mode.

wjCalendar Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Calendar** control.

Use the **wjCalendar** binding to add **Calendar** controls to your KnockoutJS applications. For example:

```
<p>Here is a Calendar control:</p>
<div
  data-bind="wjCalendar: { value: theDate }">
</div>
```

The **wjCalendar** binding supports all read-write properties and events of the **Calendar** control. The following properties provide two-way binding mode:

- **value**
- **displayMonth**

wjCollectionViewNavigator Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for an **ICollectionView** navigator element.

Use the **wjCollectionViewNavigator** directive to add an element that allows users to navigate through the items in an **ICollectionView**. For example:

```
Here is a CollectionViewNavigator:</p>
<div
  data-bind="wjCollectionViewNavigator: { cv: myCollectionView }">
</div>
```

The **wjCollectionViewNavigator** directive has a single attribute:

cv
Reference to the **ICollectionView** object to navigate.

wjCollectionViewPager Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for an **ICollectionView** pager element.

Use the **wjCollectionViewPager** directive to add an element that allows users to navigate through the pages in a paged **ICollectionView**. For example:

```
Here is a CollectionViewPager:</p>
<div
  data-bind="wjCollectionViewPager: { cv: myCollectionView }">
</div>
```

The **wjCollectionViewPager** directive has a single attribute:

cv
Reference to the paged **ICollectionView** object to navigate.

wjColorPicker Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **ColorPicker** control.

Use the **wjColorPicker** binding to add **ColorPicker** controls to your KnockoutJS applications. For example:

```
<p>Here is a ColorPicker control:</p>
<div
  data-bind="wjColorPicker: { value: theColor }">
</div>
```

The **wjColorPicker** binding supports all read-write properties and events of the **ColorPicker** control. The following properties provide two-way binding mode:

- **value**

wjComboBox Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Derived Classes
wjAutoComplete, **wjInputTime**, **wjMenu**, **wjMultiSelect**

KnockoutJS binding for the **ComboBox** control.

Use the **wjComboBox** binding to add **ComboBox** controls to your KnockoutJS applications. For example:

```
<p>Here is a ComboBox control:</p>
<div data-bind="wjComboBox: {
  itemsSource: countries,
  text: theCountry,
  isEditable: false,
  placeholder: 'country' }">
</div>
```

The **wjComboBox** binding supports all read-write properties and events of the **ComboBox** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjContextMenu Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for context menus.

Use the **wjContextMenu** binding to add context menus to elements on the page. The **wjContextMenu** binding is based on the **wjMenu**; it displays a popup menu when the user performs a context menu request on an element (usually a right-click).

The wjContextMenu binding is specified as a parameter added to the element that the context menu applies to. The parameter value is a selector for the element that contains the menu. For example:

```
<!-- paragraph with a context menu -->
<p data-bind="wjContextMenu: { id: '#idMenu'}" >
 This paragraph has a context menu.</p>

<!-- define the context menu (hidden and with an id) -->
<div id="contextmenu" data-bind="wjMenu: { header: 'File', itemClicked: menuItemClicked}">
    <span data-bind="wjMenuItem: {}">New</span>
    <span data-bind="wjMenuItem: {}">open an existing file or folder</span>
    <span data-bind="wjMenuItem: {}">save the current file</span>
    <span data-bind="wjMenuSeparator: {}"></span>
    <span data-bind="wjMenuItem: {}">exit the application</span>
</div>
```

wjFinancialChart Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FinancialChart** control.

Use the **wjFinancialChart** binding to add **FinancialChart** controls to your KnockoutJS applications. For example:

```
<p>Here is a FinancialChart control:</p>
<div data-bind="wjFinancialChart: { itemsSource: data, chartType: 'Candlestick' }">
    <div data-bind="wjFlexChartLegend : {
        position: 'Top' }">
    </div>
    <div data-bind="wjFinancialChartSeries: {
         name: 'close',
        binding: 'high,low,open,close' }">
    </div>
    </div>
</div>
```

The **wjFinancialChart** binding may contain the following child bindings: **wjFlexChartAxis**, **wjFinancialChartSeries**, **wjFlexChartLegend**.

The **wjFinancialChart** binding supports all read-write properties and events of the **FinancialChart** control, and the additional **tooltipContent** property that assigns a value to the **FinancialChart.tooltip.content** property. The **selection** property provides two-way binding mode.

wjFinancialChartSeries Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FinancialChart FinancialSeries** object.

The **WjFinancialChartSeries** binding must be contained in a **wjFinancialChart** binding.

The **WjFinancialChartSeries** binding supports all read-write properties and events of the **FinancialSeries** class. The **visibility** property provides two-way binding mode.

wjFlexChart Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexChart** control.

Use the **wjFlexChart** binding to add **FlexChart** controls to your KnockoutJS applications. For example:

```
<p>Here is a FlexChart control:</p>
<div data-bind="wjFlexChart: { itemsSource: data }">
    <div data-bind="wjFlexChartLegend : {
        position: 'Top' }">
    </div>
    <div data-bind="wjFlexChartAxis: {
        wjProperty: 'axisX',
        title: chartProps.titleX }">
    </div>
    <div data-bind="wjFlexChartAxis: {
        wjProperty: 'axisY',
        majorUnit: 5000 }">
    </div>
    <div data-bind="wjFlexChartSeries: {
        name: 'Sales',
        binding: 'sales' }">
    </div>
    <div data-bind="wjFlexChartSeries: {
        name: 'Expenses',
        binding: 'expenses' }">
    </div>
    <div data-bind="wjFlexChartSeries: {
        name: 'Downloads',
        binding: 'downloads',
        chartType: 'LineSymbols' }">
    </div>
</div>
```

The **wjFlexChart** binding may contain the following child bindings: **wjFlexChartAxis**, **wjFlexChartSeries**, **wjFlexChartLegend**.

The **wjFlexChart** binding supports all read-write properties and events of the **FlexChart** control, and the additional **tooltipContent** property that assigns a value to the **FlexChart.tooltip.content** property. The **selection** property provides two-way binding mode.

wjFlexChartAnimation Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **ChartAnimation** object.

Use the **wjFlexChartAnimation** binding to add **ChartAnimation** object to your KnockoutJS applications. For example:

```
<p>Here is a ChartAnimation:</p>
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country',chartType:'Column' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartAnimation: { animationMode: 'Series',easing:'Swing',duration:2000 }  "></div>
</div>
```

The **wjFlexChartAnimation** binding supports all read-write properties and events of the **ChartAnimation** class.

wjFlexChartAnnotation Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for annotations.

The **wjFlexChartAnnotation** must be contained in a **wjFlexChartAnnotationLayer** binding.For example:

```
<p>Here is a AnnotationLayer:</p>
<div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
   <div data-bind="wjFinancialChartSeries: { bindingX: 'date', binding: 'high,low,open,close' }"></div>
   <div data-bind="wjFlexChartAnnotationLayer: {}">
       <div data-bind="wjFlexChartAnnotation: { type: 'Rectangle', content: 'E',height:20, width:20,attachment:'DataIndex',pointIndex: 10}"></div>
       <div data-bind="wjFlexChartAnnotation: { type: 'Ellipse', content: 'E',height:20, width:20,attachment:'DataIndex',pointIndex: 30}"></div>
   </div>
      </div>
```

The **wjFlexChartAnnotation** is used to represent all types of possible annotation shapes like **Circle**, **Rectangle**, **Polygon** and so on. The type of annotation shape is specified in the **type** attribute.

wjFlexChartAnnotationLayer Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **AnnotationLayer** object.

Use the **wjFlexChartAnnotationLayer** binding to add **AnnotationLayer** object to your KnockoutJS applications. For example:

```
<p>Here is a AnnotationLayer:</p>
<div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
   <div data-bind="wjFinancialChartSeries: { bindingX: 'date', binding: 'high,low,open,close' }"></div>
   <div data-bind="wjFlexChartAnnotationLayer: {}">
       <div data-bind="wjFlexChartAnnotation: { type: 'Rectangle', content: 'E',height:20, width:20,attachment:'DataIndex',pointIndex: 10}"></div>
       <div data-bind="wjFlexChartAnnotation: { type: 'Ellipse', content: 'E',height:20, width:20,attachment:'DataIndex',pointIndex: 30}"></div>
   </div>
      </div>
```

The **wjFlexChartAnnotationLayer** binding supports all read-write properties and events of the **AnnotationLayer** class.

wjFlexChartAtr Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **ATR** object.

Use the **wjFlexChartAtr** binding to add **ATR** object to your KnockoutJS applications. For example:

```
<p>Here is a ATR:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartAtr: { binding: 'high,low,open,close',period:'14' }"></div>
  </div>
```

The **wjFlexChartAtr** binding supports all read-write properties and events of the **ATR** class.

wjFlexChartAxis Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexChart Axis** object.

The **wjFlexChartAxis** binding must be contained in a **wjFlexChart** binding. Use the **wjProperty** attribute to specify the property (**axisX** or **axisY**) to initialize with this binding.

The **wjFlexChartAxis** binding supports all read-write properties and events of the **Axis** class.

wjFlexChartBollingerBands Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **BollingerBands** object.

Use the **wjFlexChartBollingerBands** binding to add **BollingerBands** object to your KnockoutJS applications. For example:

```
<p>Here is a BollingerBands:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartStochastic: { binding: 'high,low,open,close',kPeriod:14,dPeriod:3,smoothingPeriod: 1 }" ></div>
  </div>
```

The **wjFlexChartBollingerBands** binding supports all read-write properties and events of the **BollingerBands** class.

wjFlexChartCci Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **CCI** object.

Use the **wjFlexChartCci** binding to add **CCI** object to your KnockoutJS applications. For example:

```
<p>Here is a CCI:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartCci: { binding: 'high,low,open,close',period:20 }"></div>
  </div>
```

The **wjFlexChartCci** binding supports all read-write properties and events of the **CCI** class.

wjFlexChartDataPoint Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **DataPoint** object. The **wjFlexChartDataPoint** must be contained in a **wjFlexChartAnnotation**. The property of the parent object where **wjFlexChartDataPoint** should assign a value is specified in the **wjProperty** attribute.

Use the **wjFlexChartDataPoint** binding to add **DataPoint** object to your KnockoutJS applications. For example:

```
<p>Here is a DataPoint:</p>
  <div data-bind="wjFlexChartDataPoint: { wjProperty: 'point', x: 0.9, y:0.4}" ></div>
```

The **wjFlexChartDataPoint** binding supports all read-write properties and events of the **DataPoint** class.

wjFlexChartEnvelopes Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Envelopes** object.

Use the **wjFlexChartEnvelopes** binding to add **Envelopes** object to your KnockoutJS applications. For example:

```
<p>Here is a Envelopes:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartEnvelopes: { binding:'close', type:'Simple', size: 0.03, period:20}" ></div>
  </div>
```

The **wjFlexChartEnvelopes** binding supports all read-write properties and events of the **Envelopes** class.

wjFlexChartFibonacci Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Fibonacci** object.

Use the **wjFlexChartFibonacci** binding to add **Fibonacci** object to your KnockoutJS applications. For example:

```
<p>Here is a Fibonacci:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries: { bindingX: 'date', binding: 'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacci: { binding:'close', symbolSize:1, labelPosition: 'Left',  uptrend: true}"></div>
  </div>
```

The **wjFlexChartFibonacci** binding supports all read-write properties and events of the **Fibonacci** class.

wjFlexChartFibonacciArcs Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FibonacciArcs** object.

Use the **wjFlexChartFibonacciArcs** binding to add **FibonacciArcs** object to your KnockoutJS applications. For example:

```
<p>Here is a FibonacciArcs:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries: { bindingX: 'date', binding: 'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacciArcs: { binding:'close', start:start, end: end,  labelPosition: 'Top'}"></div>
  </div>
```

The **wjFlexChartFibonacciArcs** binding supports all read-write properties and events of the **FibonacciArcs** class.

wjFlexChartFibonacciFans Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FibonacciFans** object.

Use the **wjFlexChartFibonacciFans** binding to add **FibonacciFans** object to your KnockoutJS applications. For example:

```
<p>Here is a FibonacciFans:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries: { bindingX: 'date', binding: 'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacciFans: { binding:'close', start:start, end: end,  labelPosition: 'Top'}"></div>
  </div>
```

The **wjFlexChartFibonacciFans** binding supports all read-write properties and events of the **FibonacciFans** class.

wjFlexChartFibonacciTimeZones Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FibonacciTimeZones** object.

Use the **wjFlexChartFibonacciTimeZones** binding to add **FibonacciTimeZones** object to your KnockoutJS applications. For example:

```
<p>Here is a FibonacciTimeZones:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
        <div data-bind="wjFinancialChartSeries: { bindingX: 'date', binding: 'high,low,open,close' }"></div>
        <div data-bind="wjFlexChartFibonacciTimeZones: { binding:'close', startX:zStart, endX: zEnd,  labelPosition: 'Right'}"></div>
  </div>
```

The **wjFlexChartFibonacciTimeZones** binding supports all read-write properties and events of the **FibonacciTimeZones** class.

wjFlexChartGestures Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **ChartGestures** object.

Use the **wjFlexChartGestures** binding to add **ChartGestures** controls to your KnockoutJS applications. For example:

```
<p>Here is a ChartGestures:</p>
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartGestures: { scaleX:0.5, posX:0.1 } "></div>
</div>
```

The **wjFlexChartGestures** binding supports all read-write properties and events of the **ChartGestures** class.

wjFlexChartLegend Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the Charts' **Legend** object.

The **wjFlexChartLegend** binding must be contained in one the following bindings: **wjFlexChart**, **wjFlexPie**.

The **wjFlexChartLegend** binding supports all read-write properties and events of the **Legend** class.

wjFlexChartLineMarker Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **LineMarker** control.

Use the **wjFlexChartLineMarker** binding to add **LineMarker** controls to your KnockoutJS applications. For example:

```
<p>Here is a LineMarker:</p>
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Expenses', binding: 'expenses' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Downloads', binding: 'downloads' }"></div>
    <div data-bind="wjFlexChartLineMarker: { interaction: 'Move', lines: 'Both' }"></div>
</div>
```

The **wjFlexChartLineMarker** binding supports all read-write properties and events of the **LineMarker** class.

wjFlexChartMacd Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Macd** object.

Use the **wjFlexChartMacd** binding to add **Macd** object to your KnockoutJS applications. For example:

```
<p>Here is a Macd:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartMacd: { binding: 'close',fastPeriod:12, slowPeriod: 26,smoothingPeriod: 9 }" ></div>
  </div>
```

The **wjFlexChartMacd** binding supports all read-write properties and events of the **Macd** class.

wjFlexChartMacdHistogram Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **MacdHistogram** object.

Use the **wjFlexChartMacdHistogram** binding to add **MacdHistogram** object to your KnockoutJS applications. For example:

```
<p>Here is a MacdHistogram:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="WjFlexChartMacdHistogram: { binding: 'close',fastPeriod:12, slowPeriod: 26,smoothingPeriod: 9 }" ></div>
  </div>
```

The **wjFlexChartMacdHistogram** binding supports all read-write properties and events of the **MacdHistogram** class.

wjFlexChartMovingAverage Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **MovingAverage** object.

Use the **wjFlexChartMovingAverage** binding to add **MovingAverage** object to your KnockoutJS applications. For example:

```
<p>Here is a MovingAverage:</p>
<div data-bind="wjFlexChart: { itemsSource: trendItemsSource, bindingX: 'x' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { chartType: 'Scatter', name: 'Base Data', binding: 'y' }"></div>
    <div data-bind="wjFlexChartMovingAverage: { binding: 'y', bindingX: 'x', period:2 }  "></div>
</div>
```

The **wjFlexChartMovingAverage** binding supports all read-write properties and events of the **MovingAverage** class.

wjFlexChartParametricFunctionSeries Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **ParametricFunctionSeries** object.

Use the **wjFlexChartParametricFunctionSeries** binding to add **ParametricFunctionSeries** object to your KnockoutJS applications. For example:

```
<p>Here is a ParametricFunctionSeries:</p>
<div data-bind="wjFlexChart: { itemsSource: trendItemsSource, bindingX: 'x' }">
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartParametricFunctionSeries: {  sampleCount:1000, max: max,xFunc:xFunc,yFunc:yFunc  }"></div>
</div>
```

The **wjFlexChartParametricFunctionSeries** binding supports all read-write properties and events of the **ParametricFunctionSeries** class.

wjFlexChartPlotArea Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **PlotArea** object.

Use the **wjFlexChartPlotArea** binding to add **PlotArea** object to your KnockoutJS applications. For example:

```
<p>Here is a PlotArea:</p>
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartPlotArea: { row:0, name:'plot1', style:{ fill: 'rgba(136,189,230,0.2)'} }  "></div>
</div>
```

The **wjFlexChartPlotArea** binding supports all read-write properties and events of the **PlotArea** class.

wjFlexChartRangeSelector Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **RangeSelector** control.

Use the **wjFlexChartRangeSelector** binding to add **RangeSelector** controls to your KnockoutJS applications. For example:

```
<p>Here is a RangeSelector control:</p>
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Expenses', binding: 'expenses' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Downloads', binding: 'downloads' }"></div>
    <div data-bind="wjFlexChartRangeSelector: { seamless: 'true',rangeChanged: rangeChanged }"></div>
</div>
```

The **wjFlexChartRangeSelector** binding supports all read-write properties and events of the **RangeSelector** class.

wjFlexChartRsi Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **RSI** object.

Use the **wjFlexChartRsi** binding to add **RSI** object to your KnockoutJS applications. For example:

```
<p>Here is a RSI:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date', chartType:'Candlestick' }">
        <div data-bind="wjFlexChartRsi: { binding: 'high,low,open,close',period:20 }"></div>
  </div>
```

The **wjFlexChartRsi** binding supports all read-write properties and events of the **RSI** class.

wjFlexChartSeries Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexChart Series** object.

The **wjFlexChartSeries** binding must be contained in a **wjFlexChart** binding.

The **wjFlexChartSeries** binding supports all read-write properties and events of the **Series** class. The **visibility** property provides two-way binding mode.

wjFlexChartStochastic Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Stochastic** object.

Use the **wjFlexChartStochastic** binding to add **Stochastic** object to your KnockoutJS applications. For example:

```
<p>Here is a Stochastic:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartStochastic: { binding: 'high,low,open,close',kPeriod:14,dPeriod:3,smoothingPeriod: 1 }" ></div>
  </div>
```

The **wjFlexChartStochastic** binding supports all read-write properties and events of the **Stochastic** class.

wjFlexChartTrendLine Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **TrendLine** object.

Use the **wjFlexChartTrendLine** binding to add **TrendLine** object to your KnockoutJS applications. For example:

```
<p>Here is a TrendLine:</p>
<div data-bind="wjFlexChart: { itemsSource: data, bindingX: 'country',chartType:'Column' }">
    <div data-bind="wjFlexChartAxis: { wjProperty: 'axisX', title: 'country' }"></div>
    <div data-bind="wjFlexChartSeries: { name: 'Sales', binding: 'sales' }"></div>
    <div data-bind="wjFlexChartAnimation: { animationMode: 'Series',easing:'Swing',duration:2000 }  "></div>
</div>
```

The **wjFlexChartTrendLine** binding supports all read-write properties and events of the **TrendLine** class.

wjFlexChartWaterfall Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Waterfall** object.

Use the **wjFlexChartWaterfall** binding to add **Waterfall** object to your KnockoutJS applications. For example:

```
<p>Here is a Waterfall:</p>
<div data-bind="wjFlexChart: { itemsSource: trendItemsSource,  binding:'value',bindingX: 'name' }">
    <div data-bind="wjFlexChartWaterfall: {  relativeData:true, connectorLines: true, start:1000,showIntermediateTotal: true,
                    intermediateTotalPositions: [3, 6, 9, 12], intermediateTotalLabels: ['Q1', 'Q2', 'Q3', 'Q4'],name:'Increase,Decrease,Total'}"></div>
</div>
```

The **wjFlexChartWaterfall** binding supports all read-write properties and events of the **Waterfall** class.

wjFlexChartWilliamsR Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **WilliamsR** object.

Use the **wjFlexChartWilliamsR** binding to add **WilliamsR** object to your KnockoutJS applications. For example:

```
<p>Here is a WilliamsR:</p>
   <div data-bind="wjFinancialChart: { itemsSource: fData, bindingX: 'date'}">
        <div data-bind="wjFlexChartWilliamsR: { binding: 'high,low,open,close',period:20 }"></div>
  </div>
```

The **wjFlexChartWilliamsR** binding supports all read-write properties and events of the **WilliamsR** class.

wjFlexChartYFunctionSeries Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **YFunctionSeries** object.

Use the **wjFlexChartYFunctionSeries** binding to add **YFunctionSeries** object to your KnockoutJS applications. For example:

```
<p>Here is a YFunctionSeries:</p>
<div data-bind="wjFlexChart: { itemsSource: trendItemsSource, bindingX: 'x' }">
    <div data-bind="wjFlexChartYFunctionSeries: {  min: 10, max: -10, sampleCount:100,func:func }"></div>
</div>
```

The **wjFlexChartYFunctionSeries** binding supports all read-write properties and events of the **YFunctionSeries** class.

wjFlexGrid Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Derived Classes
wjFlexSheet, **wjMultiRow**, **wjPivotGrid**

KnockoutJS binding for the **FlexGrid** control.

Use the **wjFlexGrid** binding to add **FlexGrid** controls to your KnockoutJS applications. For example:

```
<p>Here is a FlexGrid control:</p>
<div data-bind="wjFlexGrid: { itemsSource: data }">
    <div data-bind="wjFlexGridColumn: {
        header: 'Country',
        binding: 'country',
        width: '*' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Date',
        binding: 'date' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Revenue',
        binding: 'amount',
        format: 'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Active',
        binding: 'active' }">
    </div>
</div>
```

The **wjFlexGrid** binding may contain **wjFlexGridColumn** child bindings.

The **wjFlexGrid** binding supports all read-write properties and events of the **FlexGrid** control, except for the **scrollPosition**, **selection** and **columnLayout** properties.

wjFlexGridColumn Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexGrid Column** object.

The **wjFlexGridColumn** binding must be contained in a **wjFlexGrid** binding. For example:

```
<p>Here is a FlexGrid control:</p>
<div data-bind="wjFlexGrid: { itemsSource: data }">
    <div data-bind="wjFlexGridColumn: {
        header: 'Country',
        binding: 'country',
        width: '*' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Date',
        binding: 'date' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Revenue',
        binding: 'amount',
        format: 'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Active',
        binding: 'active' }">
    </div>
</div>
```

The **wjFlexGridColumn** binding supports all read-write properties and events of the **Column** class. The **isSelected** property provides two-way binding mode.

In addition to regular attributes that match properties in the **Column** class, an element with the **wjFlexGridColumn** binding may contain a **wjStyle** binding that provides conditional formatting and an HTML fragment that is used as a cell template. Grid rows automatically stretch vertically to fit custom cell contents.

Both the **wjStyle** binding and the HTML fragment can use the **$item** observable variable in KnockoutJS bindings to refer to the item that is bound to the current row. Also available are the **$row** and **$col** observable variables containing cell row and column indexes. For example:

```
<div data-bind="wjFlexGridColumn: {
        header: 'Symbol',
        binding: 'symbol',
        readOnly: true,
        width: '*' }">
  <a data-bind="attr: {
        href: 'https://finance.yahoo.com/q?s=' + $item().symbol() },
        text: $item().symbol">
  </a>
</div>
<div data-bind="wjFlexGridColumn: {
     header: 'Change',
        binding: 'changePercent',
        format: 'p2',
        width: '*'
        },
        wjStyle: {
        color: getAmountColor($item().change) }">
</div>
```

These bindings create two columns. The first has a template that produces a hyperlink based on the bound item's "symbol" property. The second has a conditional style that renders values with a color determined by a function implemented in the controller.

wjFlexGridFilter Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexGrid FlexGridFilter** object.

The **wjFlexGridFilter** binding must be contained in a **wjFlexGrid** binding. For example:

```
<p>Here is a FlexGrid control with column filters:</p>
<div data-bind="wjFlexGrid: { itemsSource: data }">
    <div data-bind="wjFlexGridFilter: { filterColumns: ['country', 'amount']  }"></div>

    <div data-bind="wjFlexGridColumn: {
        header: 'Country',
        binding: 'country',
        width: '*' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Date',
        binding: 'date' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Revenue',
        binding: 'amount',
        format: 'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Active',
        binding: 'active' }">
    </div>
</div>
```

The **wjFlexGridFilter** binding supports all read-write properties and events of the **FlexGridFilter** class.

wjFlexPie Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexPie** control.

Use the **wjFlexPie** binding to add **FlexPie** controls to your KnockoutJS applications. For example:

```
<p>Here is a FlexPie control:</p>
<div data-bind="wjFlexPie: {
        itemsSource: data,
        binding: 'value',
        bindingName: 'name',
        header: 'Fruit By Value' }">
    <div data-bind="wjFlexChartLegend : { position: 'Top' }"></div>
</div>
```

The **wjFlexPie** binding may contain the **wjFlexChartLegend** child binding.

The **wjFlexPie** binding supports all read-write properties and events of the **FlexPie** control.

wjFlexSheet Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjFlexGrid

KnockoutJS binding for the **FlexSheet** control.

Use the **wjFlexSheet** binding to add **FlexSheet** controls to your KnockoutJS applications.

The **wjFlexSheet** binding may contain **wjSheet** child bindings.

The **wjFlexSheet** binding supports all read-write properties and events of the **FlexSheet** control.

wjGroupPanel Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexGrid GroupPanel** control.

The **wjGroupPanel** binding should be connected to the **FlexGrid** control using the **grid** property. For example:

```
<p>Here is a FlexGrid control with GroupPanel:</p>

<div data-bind="wjGroupPanel: { grid: flex(), placeholder: 'Drag columns here to create groups.' }"></div>

<div data-bind="wjFlexGrid: { control: flex, itemsSource: data }">
    <div data-bind="wjFlexGridColumn: {
        header: 'Country',
        binding: 'country',
        width: '*' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Date',
        binding: 'date' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Revenue',
        binding: 'amount',
        format: 'n0' }">
    </div>
    <div data-bind="wjFlexGridColumn: {
        header: 'Active',
        binding: 'active' }">
    </div>
</div>
```

The **wjGroupPanel** binding supports all read-write properties and events of the **GroupPanel** class.

wjInputColor Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **InputColor** control.

Use the **wjInputColor** binding to add **InputColor** controls to your KnockoutJS applications. For example:

```
<p>Here is a InputColor control:</p>
<div
  data-bind="wjInputColor: { value: theColor }">
</div>
```

The **wjInputColor** binding supports all read-write properties and events of the **InputColor** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **value**

wjInputDate Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **InputDate** control.

Use the **wjInputDate** binding to add **InputDate** controls to your KnockoutJS applications. For example:

```
<p>Here is an InputDate control:</p>
<div data-bind="wjInputDate: {
  value: theDate,
  format: 'M/d/yyyy' }">
</div>
```

The **wjInputDate** binding supports all read-write properties and events of the **InputDate** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **value**

wjInputDateTime Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **InputDateTime** control.

Use the **wjInputDateTime** binding to add **InputDateTime** controls to your KnockoutJS applications.

The **wjInputDateTime** binding supports all read-write properties and events of the **InputDateTime** control.

wjInputMask Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **InputMask** control.

Use the **wjInputMask** binding to add **InputMask** controls to your KnockoutJS applications. For example:

```
<p>Here is an InputMask control:</p>
<div data-bind="wjInputMask: {
  mask: '99/99/99',
  promptChar: '*' }">
</div>
```

The **wjInputMask** binding supports all read-write properties and events of the **InputMask** control. The **value** property provides two-way binding mode.

wjInputNumber Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **InputNumber** control.

Use the **wjInputNumber** binding to add **InputNumber** controls to your KnockoutJS applications. For example:

```
<p>Here is an InputNumber control:</p>
<div data-bind="wjInputNumber: {
  value: theNumber,
  min: 0,
  max: 10,
  format: 'n0',
  placeholder: 'number between zero and ten' }">
</div>
```

The **wjInputNumber** binding supports all read-write properties and events of the **InputNumber** control. The following properties provide two-way binding mode:

- **value**
- **text**

wjInputTime Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjComboBox

KnockoutJS binding for the **InputTime** control.

Use the **wjInputTime** binding to add **InputTime** controls to your KnockoutJS applications. For example:

```
<p>Here is an InputTime control:</p>
<div data-bind="wjInputTime: {
  min: new Date(2014, 8, 1, 9, 0),
  max: new Date(2014, 8, 1, 17, 0),
  step: 15,
  format: 'h:mm tt',
  value: theDate }">
</div>
```

The **wjInputTime** binding supports all read-write properties and events of the **InputTime** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**
- **value**

wjLinearGauge Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Derived Classes
wjBulletGraph

KnockoutJS binding for the **LinearGauge** control.

Use the **wjLinearGauge** binding to add **LinearGauge** controls to your KnockoutJS applications. For example:

```
<p>Here is a LinearGauge control:</p>
<div data-bind="wjLinearGauge: {
        value: props.value,
        min: props.min,
        max: props.max,
        format: props.format,
        showRanges: props.showRanges }"
        <class="linear-gauge">
    <div data-bind="wjRange: {
            wjProperty: 'pointer',
            thickness: props.ranges.pointerThickness }">
    </div>
    <div data-bind="wjRange: {
            min: props.ranges.lower.min,
            max: props.ranges.lower.max,
            color: props.ranges.lower.color }">
    </div>
    <div data-bind="wjRange: {
            min: props.ranges.middle.min,
            max: props.ranges.middle.max,
            color: props.ranges.middle.color }">
    </div>
    <div data-bind="wjRange: {
            min: props.ranges.upper.min,
            max: props.ranges.upper.max,
            color: props.ranges.upper.color }">
    </div>
</div>
```

The **wjLinearGauge** binding may contain the **wjRange** child binding.

The **wjLinearGauge** binding supports all read-write properties and events of the **LinearGauge** control. The **value** property provides two-way binding mode.

wjListBox Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **ListBox** control.

Use the **wjListBox** binding to add **ListBox** controls to your KnockoutJS applications. For example:

```
<p>Here is a ListBox control:</p>
<div data-bind="wjListBox: {
  itemsSource: countries,
  selectedItem: theCountry }">
</div>
```

The **wjListBox** binding supports all read-write properties and events of the **ListBox** control. The following properties provide two-way binding mode:

- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjMenu Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjComboBox

KnockoutJS binding for the **Menu** control.

Use the **wjMenu** binding to add **Menu** controls to your KnockoutJS applications. For example:

```
<p>Here is a Menu control used as a value picker:</p>
<div data-bind="wjMenu: { value: tax, header: 'Tax' }">
    <span data-bind="wjMenuItem: { value: 0 }">Exempt</span>
    <span data-bind="wjMenuSeparator: {}"></span>
    <span data-bind="wjMenuItem: { value: .05 }">5%</span>
    <span data-bind="wjMenuItem: { value: .1 }">10%</span>
    <span data-bind="wjMenuItem: { value: .15 }">15%</span>
</div>
```

The **wjMenu** binding may contain the following child bindings: **wjMenuItem**, **wjMenuSeparator**.

The **wjMenu** binding supports all read-write properties and events of the **Menu** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**
- **value**

wjMenuItem Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for menu items.

Use the **wjMenuItem** binding to add menu items to a **Menu** control. The **wjMenuItem** binding must be contained in a **wjMenu** binding. For example:

```
<p>Here is a Menu control with four menu items:</p>
<div data-bind="wjMenu: { value: tax, header: 'Tax' }">
    <span data-bind="wjMenuItem: { value: 0 }">Exempt</span>
    <span data-bind="wjMenuItem: { value: .05 }">5%</span>
    <span data-bind="wjMenuItem: { value: .1 }">10%</span>
    <span data-bind="wjMenuItem: { value: .15 }">15%</span>
</div>
```

The **wjMenuItem** binding supports the following attributes:

cmd
Function to execute in the controller when the item is clicked.
cmdParam
Parameter passed to the **cmd** function when the item is clicked.
value
Value selected when the item is clicked (use either this or **cmd**).

wjMenuSeparator Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for menu separators.

The the **wjMenuSeparator** adds a non-selectable separator to a **Menu** control, and has no attributes. It must be contained in a **wjMenu** binding. For example:

```
<p>Here is a Menu control with four menu items and one separator:</p>
<div data-bind="wjMenu: { value: tax, header: 'Tax' }">
    <span data-bind="wjMenuItem: { value: 0 }">Exempt</span>
    <span data-bind="wjMenuSeparator: {}"></span>
    <span data-bind="wjMenuItem: { value: .05 }">5%</span>
    <span data-bind="wjMenuItem: { value: .1 }">10%</span>
    <span data-bind="wjMenuItem: { value: .15 }">15%</span>
</div>
```

wjMultiAutoComplete Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjAutoComplete

KnockoutJS binding for the **MultiAutoComplete** control.

Use the **wjMultiAutoComplete** binding to add **MultiAutoComplete** controls to your KnockoutJS applications. For example:

```
<p>Here is a MultiAutoComplete control:</p>
<div data-bind="MultiAutoComplete: {
  itemsSource: countries,
  maxSelectedItems: 4,}">
</div>
```

The **wjMultiAutoComplete** binding supports all read-write properties and events of the **MultiAutoComplete** control.

wjMultiRow Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjFlexGrid

KnockoutJS binding for the **MultiRow** object. Use the **wjMultiRow** binding to add **MultiRow** controls to your KnockoutJS applications. For example: The **wjMultiRow** binding supports all read-write properties and events of the **MultiRow** class.

wjMultiSelect Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjComboBox

KnockoutJS binding for the **MultiSelect** control.

Use the **wjMultiSelect** binding to add **MultiSelect** controls to your KnockoutJS applications. For example:

```
<p>Here is a MultiSelect control:</p>
<div data-bind="MultiSelect: {
  itemsSource: countries,
  isEditable: false,
  headerFormat: '{count} countries selected' }">
</div>
```

The **wjMultiSelect** binding supports all read-write properties and events of the **MultiSelect** control. The following properties provide two-way binding mode:

- **isDroppedDown**
- **text**
- **selectedIndex**
- **selectedItem**
- **selectedValue**

wjPivotChart Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **PivotChart** object. Use the **wjPivotChart** binding to add **PivotChart** controls to your KnockoutJS applications. For example: The **wjPivotChart** binding supports all read-write properties and events of the **PivotChart** class.

wjPivotGrid Class

File
`wijmo.knockout.js`
Module
wijmo.knockout
Base Class
wjFlexGrid

KnockoutJS binding for the **PivotGrid** object. Use the **wjPivotGrid** binding to add **PivotGrid** controls to your KnockoutJS applications. For example: The **wjPivotGrid** binding supports all read-write properties and events of the **PivotGrid** class.

wjPivotPanel Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **PivotPanel** object. Use the **wjPivotPanel** binding to add **PivotPanel** controls to your KnockoutJS applications. For example: The **wjPivotPanel** binding supports all read-write properties and events of the **PivotPanel** class.

wjPopup Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Popup** control.

Use the **wjPopup** binding to add **Popup** controls to your KnockoutJS applications. For example:

```
<p>Here is a Popup control triggered by a button:</p>
<button id="btn2" type="button">
    Click to show Popup
</button>
 <div class="popover" data-bind="wjPopup: {
      control: popup,
      owner: '#btn2',
      showTrigger: 'Click',
      hideTrigger: 'Click'}"
 >
<h3>
         Salutation
</h3>
 <div class="popover-content">
            Hello {{firstName}} {{lastName}}
 </div>
</div>
```

wjRadialGauge Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **RadialGauge** control.

Use the **wjRadialGauge** binding to add **RadialGauge** controls to your KnockoutJS applications. For example:

```
<p>Here is a RadialGauge control:</p>
<div data-bind="wjRadialGauge: {
        value: props.value,
        min: props.min,
        max: props.max,
        format: props.format,
        showRanges: props.showRanges }"
        class="radial-gauge">
    <div data-bind="wjRange: {
            wjProperty: 'pointer',
            thickness: props.ranges.pointerThickness }">
    </div>
    <div data-bind="wjRange: {
            min: props.ranges.lower.min,
            max: props.ranges.lower.max,
            color: props.ranges.lower.color }">
    </div>
    <div data-bind="wjRange: {
            min: props.ranges.middle.min,
            max: props.ranges.middle.max,
            color: props.ranges.middle.color }">
    </div>
    <div data-bind="wjRange: {
            min: props.ranges.upper.min,
            max: props.ranges.upper.max,
            color: props.ranges.upper.color }">
    </div>
</div>
```

The **wjRadialGauge** binding may contain the **wjRange** child binding.

The **wjRadialGauge** binding supports all read-write properties and events of the **RadialGauge** control. The **value** property provides two-way binding mode.

wjRange Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the Gauge's **Range** object.

The **wjRange** binding must be contained in one of the following bindings:

- **wjLinearGauge**
- **wjRadialGauge**
- **wjBulletGraph**

By default, this binding adds a **Range** object to the **ranges** collection of the Chart control. The **wjProperty** attribute allows you to specify another Chart property, for example the **pointer** property, to initialize with the binding.

The **wjRange** binding supports all read-write properties and events of the **Range** class.

wjSheet Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **FlexSheet Sheet** object.

The **wjSheet** binding must be contained in a **wjFlexSheet** binding.

The **wjSheet** binding supports all read-write properties and events of the **Sheet** class.

wjStyle Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for conditional formatting of **FlexGrid Column** cells.

Use the **wjStyle** binding together with the **wjFlexGridColumn** binding to provide conditional formatting to column cells. For example:

```
<div data-bind="wjFlexGridColumn: {
        header: 'Change',
        binding: 'changePercent',
        format: 'p2',
        width: '*'
        },
        wjStyle: { color: getAmountColor($item().change) }"></div>
```

The **wjStyle** uses the same syntax as the native KnockoutJS **style** binding. In addition to the view model properties, the following observable variables are available in binding expressions:

$item
References the item that is bound to the current row.
$row
The row index.
$col
The column index.

wjTooltip Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **Tooltip** class.

Use the **wjTooltip** binding to add tooltips to elements on the page. The **wjTooltip** supports HTML content, smart positioning, and touch.

The **wjTooltip** binding is specified on an element that the tooltip applies to. The value is the tooltip text or the id of an element that contains the text. For example:

```
<p data-bind="wjTooltip: '#fineprint'" >
    Regular paragraph content...</p>
...
<div id="fineprint" style="display:none" >
  <h3>Important Note</h3>
  <p>
    Data for the current quarter is estimated by pro-rating etc...</p>
</div>
```

wjTreeView Class

File
`wijmo.knockout.js`
Module
wijmo.knockout

KnockoutJS binding for the **TreeView** object. Use the **wjTreeView** binding to add **TreeView** controls to your KnockoutJS applications. For example: The **wjTreeView** binding supports all read-write properties and events of the **TreeView** class.

wijmo/wijmo.angular2.directiveBase Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.directiveBase

Basic Wijmo for Angular 2 module containing internal common services and platform options.

wijmo.angular2.directiveBase is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjBase from 'wijmo/wijmo.angular2.directiveBase';

wjBase.WjOptions.asyncBindings = false;
```

Classes

- WjOptions

WjOptions Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.directiveBase

Exposes global options for the Wijmo for Angular 2 interop.

Properties

- asyncBindings

Properties

STATIC **asyncBindings**

Indicates whether Wijmo components update binding sources of the two-way bound properties asynchronously or synchronously.

If this property is set to true (default) then changes to the Wijmo components' properties with two-way bindings (like WjInputNumber.value) will cause the component to update a binding source property asynchronously, after the current change detection cycle is completed. Otherwise, if this property is set to false, the binding source will be updated immediately. A corresponding property change event (like WjInputNumber.valueChanged) is also triggered asynchronously or synchronously depending on this property value, after the binding source was updated.

This global setting can be changed for specific instances of Wijmo components, by assigning the component's **asyncBindings** property with a specific boolean value.

Transition to asynchronous binding source updates has happened in Wijmo version 350. Before that version, binding sources were updated immediately after the component's property change. In some cases this could lead to the **ExpressionChangedAfterItHasBeenCheckedError** exception in the applications running Angular in the debug mode. For example, if your component's property value is set to 0.12345, and you two-way bind it to the **value** property of the **WjInputNumber** component with the **format** property set to **'n2'**, the WjInputNumber immediately converts this value to 0.12. This change, in turn, causes Angular to update your component property (the source of this binding), so that its value changes from 0.12345 to 0.12. If this source update is performed synchronously then the binding source property changes its value during the same change detection cycle, which is prohibited by Angular. If Angular runs in debug mode then it executes a special check after every change detection cycle, which detects this change and raises the **ExpressionChangedAfterItHasBeenCheckedError** exception. Asynchronous binding source updates resolve this problem, because the binding source property is updated after the current change detection cycle has finished.

If the **ExpressionChangedAfterItHasBeenCheckedError** is not an issue for you, and parts of you application logic are sensible to a moment when binding source update happens, you can change this functionality by setting the global **asyncBindings** property to false. This should be done before the first Wijmo component was instantiated by your application logic, and the best place to do it is the file where you declare the application's root NgModel. This can be done with the code like this:

```
import * as wjBase from 'wijmo/wijmo.angular2.directiveBase';
wjBase.WjOptions.asyncBindings = false;
```

Alternatively, you can change the update mode for the specific component using its own **asyncBindings** property. For example:

```
<wj-input-number [asyncBindings]="false" [(value)]="amount"></wj-input-number>
```

Type
boolean

wijmo/wijmo.angular2.core Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.core

Contains Angular 2 components for the **wijmo** module.

wijmo.angular2.core is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjCore from 'wijmo/wijmo.angular2.core';

@Component({
    directives: [wjCore.WjTooltip],
    template: '<span [wjTooltip]="'Greeting'">Hello</span>',
    selector: 'my-cmp',
})
export class MyCmp {
}
```

Classes

- WjComponentLoader
- WjTooltip

WjComponentLoader Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.core

TBD

WjTooltip Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.core

Angular 2 directive for the **Tooltip** class.

Use the **wjTooltip** directive to add tooltips to elements on the page. The wjTooltip directive supports HTML content, smart positioning, and touch.

The wjTooltip directive is specified as a parameter added to the element that the tooltip applies to. The parameter value is the tooltip text or the id of an element that contains the text. For example:

```
<p [wjTooltip]="'#fineprint'" >
    Regular paragraph content...</p>
...
<div id="fineprint" style="display:none">
  <h3>Important Note</h3>
  <p>
    Data for the current quarter is estimated
    by pro-rating etc.</p>
</div>
```

Properties

- initialized
- isInitialized

Methods

- created

Properties

- **initialized**

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

wijmo/wijmo.angular2.input Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input

Contains Angular 2 components for the **wijmo.input** module.

wijmo.angular2.input is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjInput from 'wijmo/wijmo.angular2.input';

@Component({
    directives: [wjInput.WjInputNumber],
    template: '<wj-input-number [(value)]="amount"></wj-input-number>',
    selector: 'my-cmp',
})
export class MyCmp {
    amount = 0;
}
```

Classes

- WjAutoComplete
- WjCalendar
- WjCollectionViewNavigator
- WjCollectionViewPager
- WjColorPicker
- WjComboBox
- WjContextMenu
- WjInputColor
- WjInputDate
- WjInputDateTime
- WjInputMask
- WjInputNumber
- WjInputTime
- WjItemTemplate
- WjListBox
- WjMenu
- WjMenuItem
- WjMenuSeparator
- WjMultiAutoComplete
- WjMultiSelect
- WjPopup

WjAutoComplete Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
AutoComplete

Angular 2 component for the **AutoComplete** control.

Use the **wj-auto-complete** component to add **AutoComplete** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjAutoComplete** component is derived from the **AutoComplete** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- maxItems
- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- text
- textChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?: any): AutoComplete
```

Initializes a new instance of the **AutoComplete** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: **any** OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
AutoComplete
Returns
AutoComplete

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
`DropDown`
Type
boolean

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

cssMatch

Gets or sets the name of the CSS class used to highlight any parts of the content that match the search terms.

Inherited From
`AutoComplete`
Type
string

delay

Gets or sets the delay, in milliseconds, between when a keystroke occurs and when the search is performed.

Inherited From
`AutoComplete`
Type
number

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
`ComboBox`
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

itemsSourceFunction

Gets or sets a function that provides list items dynamically as the user types.

The function takes three parameters:

- the query string typed by the user
- the maximum number of items to return
- the callback function to call when the results become available

For example:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // get results from the server
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // return results to the control
    callback(response);
  });
};
```

Inherited From
AutoComplete
Type
Function

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

maxItems

Gets or sets the maximum number of items to display in the drop-down list.

Inherited From
`AutoComplete`
Type
number

minLength

Gets or sets the minimum input length to trigger auto-complete suggestions.

Inherited From
`AutoComplete`
Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

searchMemberPath

Gets or sets a string containing a comma-separated list of properties to use when searching for items.

By default, the **AutoComplete** control searches for matches against the property specified by the **displayMemberPath** property. The **searchMemberPath** property allows you to search using additional properties.

For example, the code below would cause the control to display the company name and search by company name, symbol, and country:

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

Inherited From
AutoComplete
Type
string

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
ComboBox
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'selectedValue'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
`ComboBox`
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

WjCalendar Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
Calendar

Angular 2 component for the **Calendar** control.

Use the **wj-calendar** component to add **Calendar** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjCalendar** component is derived from the **Calendar** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- controlTemplate
- displayMonth
- displayMonthChangedNg
- firstDayOfWeek
- formatDayHeaders
- formatDays
- formatItemNg
- formatMonths
- formatYear
- formatYearMonth
- gotFocusNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- lostFocusNg
- max
- min
- monthView
- rightToLeft
- selectionMode
- showHeader
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onDisplayMonthChanged
- onFormatItem
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- displayMonthChanged
- formatItem
- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): Calendar
```

Initializes a new instance of the **Calendar** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
Calendar
Returns
Calendar

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

STATIC controlTemplate

Gets or sets the template used to instantiate **Calendar** controls.

Inherited From
Calendar
Type
any

displayMonth

Gets or sets the month displayed in the calendar.

Inherited From
`Calendar`
Type
Date

displayMonthChangedNg

Angular (EventEmitter) version of the Wijmo **displayMonthChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **displayMonthChanged** Wijmo event name.

Type
EventEmitter

firstDayOfWeek

Gets or sets a value that represents the first day of the week, the one displayed in the first column of the calendar.

Setting this property to null causes the calendar to use the default for the current culture. In the English culture, the first day of the week is Sunday (0); in most European cultures, the first day of the week is Monday (1).

Inherited From
`Calendar`
Type
number

formatDayHeaders

Gets or sets the format used to display the headers above the days in month view.

The default value for this property is 'ddd'.

Inherited From
`Calendar`
Type
string

formatDays

Gets or sets the format used to display the days in month view.

The default value for this property is 'd ' (the space after the 'd' prevents the format from being interpreted as 'd', the standard format used to represent the short date pattern).

Inherited From
`Calendar`
Type
string

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

formatMonths

Gets or sets the format used to display the months in year view.

The default value for this property is 'MMM'.

Inherited From
`Calendar`
Type
string

formatYear

Gets or sets the format used to display the year above the months in year view.

The default value for this property is 'yyyy'.

Inherited From
`Calendar`
Type
string

formatYearMonth

Gets or sets the format used to display the month and year above the calendar in month view.

The default value for this property is 'y'.

Inherited From
`Calendar`
Type
string

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`Calendar`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function to customize dates in the calendar.

The formatter function can add any content to any date. It allows complete customization of the appearance and behavior of the calendar.

If specified, the function takes two parameters:

- the date being formatted
- the HTML element that represents the date

For example, the code below shows weekends with a yellow background:

```
calendar.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

Inherited From
`Calendar`
Type
Function

itemValidator

Gets or sets a validator function to determine whether dates are valid for selection.

If specified, the validator function should take one parameter representing the date to be tested, and should return false if the date is invalid and should not be selectable.

For example, the code below shows weekends in a disabled state and prevents users from selecting those dates:

```
calendar.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

Inherited From
`Calendar`
Type
Function

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

max

Gets or sets the latest date that the user can select in the calendar.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Calendar`
Type
Date

min

Gets or sets the earliest date that the user can select in the calendar.

For details about using the **`min`** and **`max`** properties, please see the **`Using the min and max properties`** topic.

Inherited From
`Calendar`
Type
Date

monthView

Gets or sets a value indicating whether the calendar displays a month or a year.

Inherited From
`Calendar`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectionMode

Gets or sets a value that indicates whether users can select days, months, or no values at all.

Inherited From
`Calendar`
Type
`DateSelectionMode`

- **showHeader**

Gets or sets a value indicating whether the control displays the header area with the current month and navigation buttons.

Inherited From
`Calendar`
Type
boolean

- **value**

Gets or sets the currently selected date.

Inherited From
`Calendar`
Type
Date

- **valueChangedNg**

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

- **wjModelProperty**

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onDisplayMonthChanged

```
onDisplayMonthChanged(e?: EventArgs): void
```

Raises the **displayMonthChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Calendar
Returns
void

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
Calendar
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Calendar
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Indicates whether to update the control layout as well as the content.

Inherited From
`Calendar`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

displayMonthChanged

Occurs after the **`displayMonth`** property changes.

Inherited From
`Calendar`
Arguments
`EventArgs`

formatItem

Occurs when an element representing a day in the calendar has been created.

This event can be used to format calendar items for display. It is similar in purpose to the **itemFormatter** property, but has the advantage of allowing multiple independent handlers.

For example, the code below uses the **formatItem** event to disable weekends so they appear dimmed in the calendar:

```
// disable Sundays and Saturdays
calendar.formatItem.addHandler(function (s, e) {
  var day = e.data.getDay();
  if (day == 0 || day == 6) {
    wijmo.addClass(e.item, 'wj-state-disabled');
  }
});
```

Inherited From
Calendar
Arguments
FormatItemEventArgs

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

lostFocus

Occurs when the control loses the focus.

Inherited From
Control
Arguments
EventArgs

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`Calendar`
Arguments
`EventArgs`

WjCollectionViewNavigator Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input

Angular 2 component for an **ICollectionView** navigator element.

Use the **wj-collection-view-navigator** component to add an element that allows users to navigate through the items in an **ICollectionView**. For details about Angular 2 markup syntax, see Angular 2 Markup. For example:

```
<wj-collection-view-navigator
  [cv]="myCollectionView">
</wj-collection-view-navigator>
```

Properties

- initialized
- isInitialized
- wjModelProperty

Methods

- created

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjCollectionViewPager Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input

Angular 2 component for an **ICollectionView** pager element.

Use the **wj-collection-view-pager** component to add an element that allows users to navigate through the pages in a paged **ICollectionView**. For details about Angular 2 markup syntax, see Angular 2 Markup. For example:

```
<wj-collection-view-pager
  [cv]="myCollectionView">
</wj-collection-view-pager>
```

Properties

- initialized
- isInitialized
- wjModelProperty

Methods

- created

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjColorPicker Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
ColorPicker

Angular 2 component for the **ColorPicker** control.

Use the **wj-color-picker** component to add **ColorPicker** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjColorPicker** component is derived from the **ColorPicker** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- controlTemplate
- gotFocusNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- lostFocusNg
- palette
- rightToLeft
- showAlphaChannel
- showColorString
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): ColorPicker
```

Initializes a new instance of the **ColorPicker** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
ColorPicker
Returns
ColorPicker

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

STATIC controlTemplate

Gets or sets the template used to instantiate **ColorPicker** controls.

Inherited From
ColorPicker
Type
any

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

palette

Gets or sets an array that contains the colors in the palette.

The palette contains ten colors, represented by an array with ten strings. The first two colors are usually white and black.

Inherited From
`ColorPicker`
Type
string[]

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showAlphaChannel

Gets or sets a value indicating whether the **`ColorPicker`** allows users to edit the color's alpha channel (transparency).

Inherited From
`ColorPicker`
Type
boolean

showColorString

Gets or sets a value indicating whether the **`ColorPicker`** shows a string representation of the current color.

Inherited From
`ColorPicker`
Type
boolean

value

Gets or sets the currently selected color.

Inherited From
`ColorPicker`
Type
string

valueChangedNg

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ColorPicker
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`ColorPicker`
Arguments
`EventArgs`

WjComboBox Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
ComboBox

Angular 2 component for the **ComboBox** control.

Use the **wj-combo-box** component to add **ComboBox** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjComboBox** component is derived from the **ComboBox** control and inherits all its properties, events and methods.

The **wj-combo-box** component may contain a **WjItemTemplate** child directive.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- placeholder
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- text
- textChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging

- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): ComboBox
```

Initializes a new instance of the **ComboBox** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
ComboBox
Returns
ComboBox

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the `itemsSource` collection.

Inherited From
`ComboBox`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ComboBox
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'selectedValue'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
`ComboBox`
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

WjContextMenu Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input

Angular 2 directive for context menus.

Use the **wjContextMenu** directive to add context menus to elements on the page. The wjContextMenu directive is based on the **wj-menu** component; it displays a popup menu when the user performs a context menu request on an element (usually a right-click).

The wjContextMenu directive is specified as a parameter added to the element that the context menu applies to. The parameter value is a reference to the **wj-menu** component. For example:

```
<!-- paragraph with a context menu -->
<p [wjContextMenu]="menu" >
 This paragraph has a context menu.</p>

<!-- define the context menu (hidden and with an id) -->
<wj-menu #menu style="display:none">
 <wj-menu-item [cmd]="cmdOpen" [cmdParam] ="1">Open...</wj-menu-item>
 <wj-menu-item [cmd]="cmdSave" [cmdParam]="2">Save </wj-menu-item>
 <wj-menu-item [cmd]="cmdSave" [cmdParam]="3">Save As...</wj-menu-item>
 <wj-menu-item [cmd]="cmdNew" [cmdParam] ="4">New...</wj-menu-item>
 <wj-menu-separator></wj-menu-separator>
 <wj-menu-item [cmd]="cmdExit" [cmdParam]="5">Exit</wj-menu-item>
</wj-menu >
```

WjInputColor Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
InputColor

Angular 2 component for the **InputColor** control.

Use the **wj-input-color** component to add **InputColor** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjInputColor** component is derived from the **InputColor** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- colorPicker
- controlTemplate
- dropDown
- dropDownCssClass
- gotFocusNg
- hostElement
- initialized
- inputElement
- isAnimated
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- lostFocusNg
- palette
- placeholder
- rightToLeft
- showAlphaChannel
- showDropDownButton
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputColor
```

Initializes a new instance of the **InputColor** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
InputColor
Returns
InputColor

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

colorPicker

Gets a reference to the **`ColorPicker`** control shown in the drop-down.

Inherited From
`InputColor`
Type
`ColorPicker`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

palette

Gets or sets an array that contains the colors in the palette.

The palette contains ten colors, represented by an array with ten strings. The first two colors are usually white and black.

Inherited From
`InputColor`
Type
string[]

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showAlphaChannel

Gets or sets a value indicating whether the **`ColorPicker`** allows users to edit the color's alpha channel (transparency).

Inherited From
`InputColor`
Type
boolean

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`InputColor`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

value

Gets or sets the current color.

Inherited From
`InputColor`
Type
string

valueChangedNg

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputColor
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputColor`
Arguments
`EventArgs`

WjInputDate Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
InputDate

Angular 2 component for the **InputDate** control.

Use the **wj-input-date** component to add **InputDate** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjInputDate** component is derived from the **InputDate** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- gotFocusNg
- hostElement
- initialized
- inputElement
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- lostFocusNg
- mask
- max
- min
- placeholder
- rightToLeft
- selectionMode
- showDropDownButton
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputDate
```

Initializes a new instance of the **InputDate** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
InputDate
Returns
InputDate

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

calendar

Gets a reference to the **`Calendar`** control shown in the drop-down box.

Inherited From
`InputDate`
Type
`Calendar`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

format

Gets or sets the format used to display the selected date.

The format string is expressed as a .NET-style **`Date format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Inherited From
`InputDate`
Type
string

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`InputDate`
Type
HTMLInputElement

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Inherited From
`InputDate`
Type
string

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function to customize dates in the drop-down calendar.

The formatter function can add any content to any date. It allows complete customization of the appearance and behavior of the calendar.

If specified, the function takes two parameters:

- the date being formatted
- the HTML element that represents the date

For example, the code below shows weekends with a yellow background:

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

Inherited From
`InputDate`
Type
Function

itemValidator

Gets or sets a validator function to determine whether dates are valid for selection.

If specified, the validator function should take one parameter representing the date to be tested, and should return false if the date is invalid and should not be selectable.

For example, the code below prevents users from selecting dates that fall on weekends:

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

Inherited From
`InputDate`
Type
Function

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mask

Gets or sets a mask to use while editing.

The mask format is the same one that the **InputMask** control uses.

If specified, the mask must be compatible with the value of the **format** property. For example, the mask '99/99/9999' can be used for entering dates formatted as 'MM/dd/yyyy'.

Inherited From
InputDate
Type
string

max

Gets or sets the latest date that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputDate
Type
Date

min

Gets or sets the earliest date that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputDate
Type
Date

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectionMode

Gets or sets a value that indicates whether users can select days, months, or no values at all.

This property affects the behavior of the drop-down calendar, but not the format used to display dates. If you set **`selectionMode`** to 'Month', you should normally set the **`format`** property to 'MMM yyyy' or some format that does not include the day. For example:

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

Inherited From
`InputDate`
Type
`DateSelectionMode`

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`InputDate`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

value

Gets or sets the current date.

Inherited From
`InputDate`
Type
Date

valueChangedNg

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputDate
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputDate`
Arguments
`EventArgs`

WjInputDateTime Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
InputDateTime

Angular 2 component for the **InputDateTime** control.

Use the **wj-input-date-time** component to add **InputDateTime** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjInputDateTime** component is derived from the **InputDateTime** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- calendar
- controlTemplate
- dropDown
- dropDownCssClass
- format
- gotFocusNg
- hostElement
- initialized
- inputElement
- inputTime
- inputType
- isAnimated
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemValidator
- lostFocusNg
- mask
- max
- min
- placeholder
- rightToLeft
- selectionMode
- showDropDownButton
- text
- textChangedNg
- timeFormat
- timeMax
- timeMin
- timeStep
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputDateTime
```

Initializes a new instance of the **InputDateTime** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
InputDateTime
Returns
InputDateTime

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
DropDown
Type
boolean

calendar

Gets a reference to the **`Calendar`** control shown in the drop-down box.

Inherited From
`InputDate`
Type
`Calendar`

STATIC controlTemplate

Gets or sets the template used to instantiate **`InputDateTime`** controls.

Inherited From
`InputDateTime`
Type
any

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

format

Gets or sets the format used to display the selected date.

The format string is expressed as a .NET-style **`Date format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Inherited From
`InputDate`
Type
string

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`InputDate`
Type
HTMLInputElement

inputTime

Gets a reference to the inner **`InputTime`** control so you can access its full object model.

Inherited From
`InputDateTime`
Type
`InputTime`

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Inherited From
`InputDate`
Type
string

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function to customize dates in the drop-down calendar.

The formatter function can add any content to any date. It allows complete customization of the appearance and behavior of the calendar.

If specified, the function takes two parameters:

- the date being formatted
- the HTML element that represents the date

For example, the code below shows weekends with a yellow background:

```
inputDate.itemFormatter = function(date, element) {
  var day = date.getDay();
  element.style.backgroundColor = day == 0 || day == 6 ? 'yellow' : '';
}
```

Inherited From
`InputDate`
Type
Function

itemValidator

Gets or sets a validator function to determine whether dates are valid for selection.

If specified, the validator function should take one parameter representing the date to be tested, and should return false if the date is invalid and should not be selectable.

For example, the code below prevents users from selecting dates that fall on weekends:

```
inputDate.itemValidator = function(date) {
  var weekday = date.getDay();
  return weekday != 0 && weekday != 6;
}
```

Inherited From
`InputDate`
Type
Function

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mask

Gets or sets a mask to use while editing.

The mask format is the same one that the **`InputMask`** control uses.

If specified, the mask must be compatible with the value of the **`format`** property. For example, the mask '99/99/9999' can be used for entering dates formatted as 'MM/dd/yyyy'.

Inherited From
`InputDate`
Type
string

max

Gets or sets the latest date that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputDate
Type
Date

min

Gets or sets the earliest date that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputDate
Type
Date

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
DropDown
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
Control
Type
boolean

selectionMode

Gets or sets a value that indicates whether users can select days, months, or no values at all.

This property affects the behavior of the drop-down calendar, but not the format used to display dates. If you set **`selectionMode`** to 'Month', you should normally set the **`format`** property to 'MMM yyyy' or some format that does not include the day. For example:

```
var inputDate = new wijmo.input.InputDate('#el, {
  selectionMode: 'Month',
  format: 'MMM yyyy'
});
```

Inherited From
`InputDate`
Type
`DateSelectionMode`

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`InputDate`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

timeFormat

Gets or sets the format used to display times in the drop-down list.

This property does not affect the value shown in the control's input element. That value is formatted using the **`format`** property.

The format string is expressed as a .NET-style **`time format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)`**.

Inherited From
`InputDateTime`
Type
string

timeMax

Gets or sets the latest time that the user can enter.

Inherited From
`InputDateTime`
Type
Date

timeMin

Gets or sets the earliest time that the user can enter.

Inherited From
`InputDateTime`
Type
Date

timeStep

Gets or sets the number of minutes between entries in the drop-down list of times.

Inherited From
`InputDateTime`
Type
number

value

Gets or sets the current date.

Inherited From
`InputDate`
Type
Date

valueChangedNg

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputDate
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputDate`
Arguments
`EventArgs`

WjInputMask Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
InputMask

Angular 2 component for the **InputMask** control.

Use the **wj-input-mask** component to add **InputMask** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjInputMask** component is derived from the **InputMask** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- controlTemplate
- gotFocusNg
- hostElement
- initialized
- inputElement
- isDisabled
- isInitialized
- isRequired
- isTouching
- isUpdating
- lostFocusNg
- mask
- maskFull
- placeholder
- promptChar
- rawValue
- rightToLeft
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- lostFocus
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputMask
```

Initializes a new instance of the **InputMask** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
InputMask
Returns
InputMask

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

STATIC controlTemplate

Gets or sets the template used to instantiate **InputMask** controls.

Inherited From
InputMask
Type
any

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`InputMask`
Type
HTMLInputElement

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isRequired

Gets or sets a value indicating whether the control value must be a non-empty string.

Inherited From
`InputMask`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mask

Gets or sets the mask used to validate the input as the user types.

The mask is defined as a string with one or more of the masking characters listed in the `InputMask` topic.

Inherited From
`InputMask`
Type
string

maskFull

Gets a value that indicates whether the mask has been completely filled.

Inherited From
`InputMask`
Type
boolean

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`InputMask`
Type
string

promptChar

Gets or sets the symbol used to show input positions in the control.

Inherited From
`InputMask`
Type
string

rawValue

Gets or sets the raw value of the control (excluding mask literals).

The raw value of the control excludes prompt and literal characters. For example, if the **`mask`** property is set to "AA-9999" and the user enters the value "AB-1234", the **`rawValue`** property will return "AB1234", excluding the hyphen that is part of the mask.

Inherited From
`InputMask`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

- **value**

Gets or sets the text currently shown in the control.

Inherited From
`InputMask`
Type
string

- **valueChangedNg**

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

- **wjModelProperty**

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputMask
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`InputMask`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputMask`
Arguments
`EventArgs`

WjInputNumber Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
InputNumber

Angular 2 component for the **InputNumber** control.

Use the **wj-input-number** component to add **InputNumber** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjInputNumber** component is derived from the **InputNumber** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- controlTemplate
- format
- gotFocusNg
- hostElement
- initialized
- inputElement
- inputType
- isDisabled
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- lostFocusNg
- max
- min
- placeholder
- rightToLeft
- showSpinner
- step
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- lostFocus
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputNumber
```

Initializes a new instance of the **InputNumber** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
InputNumber
Returns
InputNumber

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

STATIC controlTemplate

Gets or sets the template used to instantiate **InputNumber** controls.

Inherited From
InputNumber
Type
any

format

Gets or sets the format used to display the number being edited (see **`Globalize`**).

The format string is expressed as a .NET-style **`standard numeric format string (http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx)`**.

Inherited From
`InputNumber`
Type
string

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`InputNumber`
Type
HTMLInputElement

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Inherited From
`InputNumber`
Type
string

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`InputNumber`
Type
boolean

isRequired

Gets or sets a value indicating whether the control value must be a number or whether it can be set to null (by deleting the content of the control).

Inherited From
`InputNumber`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

max

Gets or sets the largest number that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputNumber
Type
number

min

Gets or sets the smallest number that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputNumber
Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
InputNumber
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showSpinner

Gets or sets a value indicating whether the control displays spinner buttons to increment or decrement the value (the step property must be set to a value other than zero).

Inherited From
`InputNumber`
Type
boolean

step

Gets or sets the amount to add or subtract to the **`value`** property when the user clicks the spinner buttons.

Inherited From
`InputNumber`
Type
number

text

Gets or sets the text shown in the control.

Inherited From
`InputNumber`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

value

Gets or sets the current value of the control.

Inherited From
`InputNumber`
Type
number

valueChangedNg

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputNumber
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`InputNumber`
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
`Control`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`InputNumber`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`InputNumber`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputNumber`
Arguments
`EventArgs`

WjInputTime Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
InputTime

Angular 2 component for the **InputTime** control.

Use the **wj-input-time** component to add **InputTime** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjInputTime** component is derived from the **InputTime** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- format
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- inputType
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- mask
- max
- maxDropDownHeight
- maxDropDownWidth
- min
- placeholder
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- step
- text
- textChangedNg
- value
- valueChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- onValueChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged
- valueChanged

Constructor

constructor

```
constructor(element: any, options?): InputTime
```

Initializes a new instance of the **InputTime** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
InputTime
Returns
InputTime

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
`DropDown`
Type
boolean

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **isDroppedDown** property is set to true.

Inherited From
DropDown
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
DropDown
Type
string

format

Gets or sets the format used to display the selected time (see **Globalize**).

The format string is expressed as a .NET-style **time format string (http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx)**.

Inherited From
InputTime
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **formatItem** event.

Inherited From
ComboBox
Type
Event

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`InputTime`
Type
HTMLInputElement

inputType

Gets or sets the "type" attribute of the HTML input element hosted by the control.

By default, this property is set to "tel", a value that causes mobile devices to show a numeric keypad that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In those cases, try changing the value to "number" or "text."

Note that input elements with type "number" prevent selection in Chrome and therefore is not recommended. For more details, see this link: http://stackoverflow.com/questions/21177489/selectionstart-selectionend-on-input-type-number-no-longer-allowed-in-chrome

Inherited From
`InputTime`
Type
string

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
`ComboBox`
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mask

Gets or sets a mask to use while the user is editing.

The mask format is the same used by the **InputMask** control.

If specified, the mask must be compatible with the value of the **format** property. For example, you can use the mask '99:99 >LL' for entering short times (format 't').

Inherited From
InputTime
Type
string

max

Gets or sets the latest time that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputTime
Type
Date

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
ComboBox
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
ComboBox
Type
number

min

Gets or sets the earliest time that the user can enter.

For details about using the **min** and **max** properties, please see the **Using the min and max properties** topic.

Inherited From
InputTime
Type
Date

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

step

Gets or sets the number of minutes between entries in the drop-down list.

The default value for this property is 15 minutes. Setting it to null, zero, or any negative value disables the drop-down.

Inherited From
`InputTime`
Type
number

text

Gets or sets the text shown in the control.

Inherited From
`InputTime`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

value

Gets or sets the current input time.

Inherited From
`InputTime`
Type
Date

valueChangedNg

Angular (EventEmitter) version of the Wijmo **valueChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **valueChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'value'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
`ComboBox`
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onValueChanged

```
onValueChanged(e?: EventArgs): void
```

Raises the **valueChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
InputTime
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

valueChanged

Occurs when the value of the **`value`** property changes, either as a result of user actions or by assignment in code.

Inherited From
`InputTime`
Arguments
`EventArgs`

WjItemTemplate Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input

Angular 2 component for the @see: control.

The **[wjItemTemplate]** directive must be contained in one of the following components: **WjListBox** , **WjMenu** , **WjComboBox** or **WjMultiSelect**.

The **[wjItemTemplate]** directive defines a template for items of a component that it's nested in. The template may contain an arbitrary HTML fragment with Angular 2 bindings and directives. The local **item**, **itemIndex** and **control** template variables can be used in Angular 2 bindings that refer to the data item, its index, and the owner control. For example:

```
<wj-list-box style="max-height:300px;width:250px;"
            [itemsSource]="musicians">
  <template wjItemTemplate let-item="item" let-itemIndex="itemIndex">
      {{itemIndex + 1}}. <b>{{item.name}}</b>
      <div *ngIf="item.photo">
          <img [src]="item.photo" height="100" />
          <br />
          <a href="https://www.google.com/#newwindow=1&q=The+Beatles+"
             target="_blank"
             style="color:red">go there!</a>
      </div>
  </template>
</wj-list-box>
```

Properties

- initialized
- isInitialized

Methods

- created

Properties

- **initialized**

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjListBox Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
ListBox

Angular 2 component for the **ListBox** control.

Use the **wj-list-box** component to add **ListBox** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjListBox** component is derived from the **ListBox** control and inherits all its properties, events and methods.

The **wj-list-box** component may contain a **WjItemTemplate** child directive.

Constructor

- constructor

Properties

- asyncBindings
- checkedItems
- checkedItemsChangedNg
- checkedMemberPath
- collectionView
- displayMemberPath
- formatItemNg
- gotFocusNg
- hostElement
- initialized
- isContentHtml
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemCheckedNg
- itemFormatter
- itemRole
- itemsChangedNg
- itemsSource
- lostFocusNg
- maxHeight
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getDisplayValue
- getItemChecked
- getTemplate
- initialize
- invalidate
- invalidateAll
- isItemEnabled
- onCheckedItemsChanged
- onFormatItem
- onGotFocus
- onItemChecked
- onItemsChanged
- onLoadedItems
- onLoadingItems
- onLostFocus
- onSelectedIndexChanged
- refresh
- refreshAll
- removeEventListener
- setItemChecked
- showSelection
- toggleItemChecked

Events

- checkedItemsChanged
- formatItem
- gotFocus
- itemChecked
- itemsChanged
- loadedItems
- loadingItems
- lostFocus
- selectedIndexChanged

Constructor

constructor

```
constructor(element: any, options?): ListBox
```

Initializes a new instance of the **ListBox** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
ListBox
Returns
ListBox

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

checkedItems

Gets or sets an array containing the items that are currently checked.

Inherited From
ListBox
Type
any[]

checkedItemsChangedNg

Angular (EventEmitter) version of the Wijmo **checkedItemsChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **checkedItemsChanged** Wijmo event name.

Type
EventEmitter

checkedMemberPath

Gets or sets the name of the property used to control the CheckBoxes placed next to each item.

Use this property to create multi-select LisBoxes. When an item is checked or unchecked, the control raises the **`itemChecked`** event. Use the **`selectedItem`** property to retrieve the item that was checked or unchecked, or use the **`checkedItems`** property to retrieve the list of items that are currently checked.

Inherited From
`ListBox`
Type

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ListBox`
Type
`ICollectionView`

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ListBox`
Type
string

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isContentHtml

Gets or sets a value indicating whether items contain plain text or HTML.

Inherited From
`ListBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemCheckedNg

Angular (EventEmitter) version of the Wijmo **itemChecked** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **itemChecked** Wijmo event name.

Type
EventEmitter

itemFormatter

Gets or sets a function used to customize the values shown on the list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
listBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ListBox
Type
Function

itemRole

Gets or sets the value or the "role" attribute added to the list items. The default value for this property is "option".

Inherited From
ListBox
Type
string

itemsChangedNg

Angular (EventEmitter) version of the Wijmo **itemsChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **itemsChanged** Wijmo event name.

Type
EventEmitter

itemsSource

Gets or sets the array or **ICollectionView** object that contains the list items.

Inherited From
ListBox
Type
any

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxHeight

Gets or sets the maximum height of the list.

Inherited From
ListBox
Type
number

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
Control
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item.

Inherited From
`ListBox`
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected.

Inherited From
`ListBox`
Type
any

selectedValue

Gets or sets the value of the `selectedItem` obtained using the `selectedValuePath`.

Inherited From
`ListBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ListBox`
Type
string

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'selectedValue'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index: number): string
```

Gets the text displayed for the item at a given index (as plain text).

Parameters

- **index**: **number**
  The index of the item.

Inherited From
`ListBox`
Returns
string

getDisplayValue

```
getDisplayValue(index: number): string
```

Gets the string displayed for the item at a given index.

The string may be plain text or HTML, depending on the setting of the **`isContentHtml`** property.

Parameters

- **index**: **number**
  The index of the item.

Inherited From
`ListBox`
Returns
string

getItemChecked

```
getItemChecked(index: number): boolean
```

Gets the checked state of an item on the list.

This method is applicable only on multi-select ListBoxes (see the **`checkedMemberPath`** property).

Parameters

- **index**: **number**
  Item index.

Inherited From
`ListBox`
Returns
boolean

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isItemEnabled

```
isItemEnabled(index: number): void
```

Gets a value that determines whether the item at a given index is enabled.

Parameters

- **index**: **number**
  The index of the item.

Inherited From
ListBox
Returns
void

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

Raises the **checkedItemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ListBox
Returns
void

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
ListBox
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onItemChecked

```
onItemChecked(e?: EventArgs): void
```

Raises the **itemChecked** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ListBox
Returns
void

onItemsChanged

```
onItemsChanged(e?: EventArgs): void
```

Raises the **itemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ListBox
Returns
void

onLoadedItems

```
onLoadedItems(e?: EventArgs): void
```

Raises the **loadedItems** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ListBox
Returns
void

onLoadingItems

```
onLoadingItems(e?: EventArgs): void
```

Raises the **`loadingItems`** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`ListBox`
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **`lostFocus`** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`Control`
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **`selectedIndexChanged`** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`ListBox`
Returns
void

refresh

```
refresh(): void
```

Refreshes the list.

Inherited From
ListBox
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

setItemChecked

```
setItemChecked(index: number, checked: boolean): void
```

Sets the checked state of an item on the list.

This method is applicable only on multi-select ListBoxes (see the **`checkedMemberPath`** property).

Parameters

- **index**: **number**
  Item index.
- **checked**: **boolean**
  Item's new checked state.

Inherited From
`ListBox`
Returns
void

showSelection

```
showSelection(): void
```

Highlights the selected item and scrolls it into view.

Inherited From
`ListBox`
Returns
void

toggleItemChecked

```
toggleItemChecked(index: number): void
```

Toggles the checked state of an item on the list. This method is applicable only to multi-select ListBoxes (see the **`checkedMemberPath`** property).

Parameters

- **index**: **number**
  Item index.

Inherited From
`ListBox`
Returns
void

Events

checkedItemsChanged

Occurs when the value of the **`checkedItems`** property changes.

Inherited From
`ListBox`
Arguments
`EventArgs`

formatItem

Occurs when an element representing a list item has been created.

This event can be used to format list items for display. It is similar in purpose to the **`itemFormatter`** property, but has the advantage of allowing multiple independent handlers.

Inherited From
`ListBox`
Arguments
`FormatItemEventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

itemChecked

Occurs when the current item is checked or unchecked by the user.

This event is raised when the **`checkedMemberPath`** is set to the name of a property to add CheckBoxes to each item in the control.

Use the **`selectedItem`** property to retrieve the item that was checked or unchecked.

Inherited From
`ListBox`
Arguments
`EventArgs`

itemsChanged

Occurs when the list of items changes.

Inherited From
`ListBox`
Arguments
`EventArgs`

loadedItems

Occurs after the list items have been generated.

Inherited From
`ListBox`
Arguments
`EventArgs`

loadingItems

Occurs before the list items are generated.

Inherited From
`ListBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ListBox`
Arguments
`EventArgs`

WjMenu Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
Menu

Angular 2 component for the **Menu** control.

Use the **wj-menu** component to add **Menu** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjMenu** component is derived from the **Menu** control and inherits all its properties, events and methods.

The **wj-menu** component may contain the following child components: **WjMenuItem** , **WjMenuSeparator** and **WjItemTemplate**.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- collectionView
- command
- commandParameterPath
- commandPath
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- header
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isButton
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemClickedNg
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- owner
- placeholder
- rightToLeft
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- text
- textChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- hide
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemClicked
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll
- show

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemClicked
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): Menu
```

Initializes a new instance of the **Menu** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
Menu
Returns
Menu

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
`DropDown`
Type
boolean

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

command

Gets or sets the command to execute when an item is clicked.

Commands are objects that implement two methods:

- **executeCommand(parameter)** This method executes the command.
- **canExecuteCommand(parameter)** This method returns a Boolean value that determines whether the controller can execute the command. If this method returns false, the menu option is disabled.

You can also set commands on individual items using the **`commandPath`** property.

Inherited From
`Menu`
Type
any

commandParameterPath

Gets or sets the name of the property that contains a parameter to use with the command specified by the **`commandPath`** property.

Inherited From
`Menu`
Type
string

commandPath

Gets or sets the name of the property that contains the command to execute when the user clicks an item.

Commands are objects that implement two methods:

- **executeCommand(parameter)** This method executes the command.
- **canExecuteCommand(parameter)** This method returns a Boolean value that determines whether the controller can execute the command. If this method returns false, the menu option is disabled.

Inherited From
`Menu`
Type
string

STATIC controlTemplate

Gets or sets the template used to instantiate `DropDown` controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the `isDroppedDown` property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

header

Gets or sets the HTML text shown in the **Menu** element.

Inherited From
Menu
Type
string

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
ComboBox
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isButton

Gets or sets a value that determines whether this **Menu** should act as a split button instead of a regular menu.

The difference between regular menus and split buttons is what happens when the user clicks the menu header. In regular menus, clicking the header shows or hides the menu options. In split buttons, clicking the header raises the **itemClicked** event and/or invokes the command associated with the last option selected by the user as if the user had picked the item from the drop-down list.

If you want to differentiate between clicks on menu items and the button part of a split button, check the value of the **isDroppedDown** property of the event sender. If that is true, then a menu item was clicked; if it is false, then the button was clicked.

For example, the code below implements a split button that uses the drop-down list only to change the default item/command, and triggers actions only when the button is clicked:

```
<-- view -->
<wj-menu is-button="true" header="Run" value="browser"
  item-clicked="itemClicked(s, e)">
  <wj-menu-item value="'Internet Explorer'">Internet Explorer</wj-menu-item>
  <wj-menu-item value="'Chrome'">Chrome</wj-menu-item>
  <wj-menu-item value="'Firefox'">Firefox</wj-menu-item>
  <wj-menu-item value="'Safari'">Safari</wj-menu-item>
  <wj-menu-item value="'Opera'">Opera</wj-menu-item>
</wj-menu>

// controller
$scope.browser = 'Internet Explorer';
$scope.itemClicked = function (s, e) {

  // if not dropped down, click was on the button
  if (!s.isDroppedDown) {
    alert('running ' + $scope.browser);
  }
}
```

Inherited From
Menu
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
ComboBox
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

- **isEditable**

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

- **isInitialized**

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

- **isReadOnly**

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

- **isRequired**

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemClickedNg

Angular (EventEmitter) version of the Wijmo **itemClicked** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **itemClicked** Wijmo event name.

Type
EventEmitter

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
ComboBox
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
ComboBox
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
ComboBox
Type
number

owner

Gets or sets the element that owns this **Menu**.

This variable is set by the wj-context-menu directive in case a single menu is used as a context menu for several different elements.

Inherited From
Menu
Type
HTMLElement

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'selectedValue'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
`ComboBox`
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hide

```
hide(): void
```

Hides the menu.

This method is useful if you want to hide a context menu displayed with the **show** method.

Inherited From
Menu
Returns
void

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.

- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemClicked

```
onItemClicked(e?: EventArgs): void
```

Raises the **itemClicked** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Menu
Returns
void

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

show

```
show(position?: any): void
```

Shows the menu at a given location.

This method is useful if you want to use the menu as a context menu, attached to one or more elements on the page. For example:

```
// create menu
var div = document.createElement('div');
var menu = new wijmo.input.Menu(div, {
    itemsSource: 'New,Open,Save,Exit'.split(','),
    itemClicked: function (s, e) {
        alert('thanks for picking ' + menu.selectedIndex);
    }
});

// use it as a context menu for one or more elements
var element = document.getElementById('btn');
element.addEventListener('contextmenu', function (e) {
    e.preventDefault();
    menu.show(e);
});
```

Parameters

- **position**: **any** OPTIONAL
  An optional **MouseEvent** or reference element that determines the position where the menu should be displayed. If not provided, the menu is displayed at the center of the screen.

Inherited From
`Menu`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemClicked

Occurs when the user picks an item from the menu.

The handler can determine which item was picked by reading the event sender's **`selectedIndex`** property.

Inherited From
`Menu`
Arguments
`EventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

WjMenuItem Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Derived Classes
WjMenuSeparator

Angular 2 component for the @see: control.

The **wj-menu-item** component must be contained in a **WjMenu** component.

Use the **wj-menu-item** component to add controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

Properties

- initialized
- isInitialized
- wjProperty

Methods

- created

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'itemsSource'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjMenuSeparator Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
WjMenuItem

Angular 2 component for the @see: control.

The **wj-menu-separator** component must be contained in a **WjMenu** component.

Use the **wj-menu-separator** component to add controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Inherited From
`WjMenuItem`
Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Inherited From
`WjMenuItem`
Type
boolean

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'itemsSource'.

Inherited From
`WjMenuItem`
Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Inherited From
`WjMenuItem`
Returns
void

WjMultiAutoComplete Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
MultiAutoComplete

Angular 2 component for the **MultiAutoComplete** control.

Use the **wj-multi-auto-complete** component to add **MultiAutoComplete** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjMultiAutoComplete** component is derived from the **MultiAutoComplete** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- collectionView
- controlTemplate
- cssMatch
- delay
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceFunction
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- maxItems
- maxSelectedItems
- minLength
- placeholder
- rightToLeft
- searchMemberPath
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedItems
- selectedItemsChangedNg
- selectedMemberPath
- selectedValue
- selectedValuePath
- showDropDownButton
- text
- textChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onSelectedItemsChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- selectedItemsChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): MultiAutoComplete
```

Initializes a new instance of the **`MultiAutoComplete`** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
MultiAutoComplete
Returns
MultiAutoComplete

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
`DropDown`
Type
boolean

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

cssMatch

Gets or sets the name of the CSS class used to highlight any parts of the content that match the search terms.

Inherited From
`AutoComplete`
Type
string

delay

Gets or sets the delay, in milliseconds, between when a keystroke occurs and when the search is performed.

Inherited From
`AutoComplete`
Type
number

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
`ComboBox`
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

itemsSourceFunction

Gets or sets a function that provides list items dynamically as the user types.

The function takes three parameters:

- the query string typed by the user
- the maximum number of items to return
- the callback function to call when the results become available

For example:

```
autoComplete.itemsSourceFunction = function (query, max, callback) {

  // get results from the server
  var params = { query: query, max: max };
  $.getJSON('companycatalog.ashx', params, function (response) {

    // return results to the control
    callback(response);
  });
};
```

Inherited From
AutoComplete
Type
Function

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
`ComboBox`
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

maxItems

Gets or sets the maximum number of items to display in the drop-down list.

Inherited From
`AutoComplete`
Type
number

maxSelectedItems

Gets or sets the maximum number of items that can be selected.

Setting this property to null (the default value) allows users to pick any number of items.

Inherited From
`MultiAutoComplete`
Type
number

minLength

Gets or sets the minimum input length to trigger auto-complete suggestions.

Inherited From
`AutoComplete`
Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

searchMemberPath

Gets or sets a string containing a comma-separated list of properties to use when searching for items.

By default, the **AutoComplete** control searches for matches against the property specified by the **displayMemberPath** property. The **searchMemberPath** property allows you to search using additional properties.

For example, the code below would cause the control to display the company name and search by company name, symbol, and country:

```
var ac = new wijmo.input.AutoComplete('#autoComplete', {
  itemsSource: companies,
  displayMemberPath: 'name',
  searchMemberPath: 'symbol,country'
});
```

Inherited From
AutoComplete
Type
string

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
ComboBox
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedItems

Gets or sets an array containing the items that are currently selected.

Inherited From
`MultiAutoComplete`
Type
any[]

selectedItemsChangedNg

Angular (EventEmitter) version of the Wijmo **selectedItemsChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedItemsChanged** Wijmo event name.

Type
EventEmitter

selectedMemberPath

Gets or sets the name of the property used to control which item will be selected.

Inherited From
`MultiAutoComplete`
Type
string

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Override the value for indicating control should not display a drop-down button.

Inherited From
`MultiAutoComplete`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'selectedItems'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
`ComboBox`
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onSelectedItemsChanged

```
onSelectedItemsChanged(e?: EventArgs): void
```

Raises the **selectedItemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`MultiAutoComplete`
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`DropDown`
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

selectedItemsChanged

Occurs when the value of the **`selectedItems`** property changes.

Inherited From
`MultiAutoComplete`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

WjMultiSelect Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
MultiSelect

Angular 2 component for the **MultiSelect** control.

Use the **wj-multi-select** component to add **MultiSelect** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjMultiSelect** component is derived from the **MultiSelect** control and inherits all its properties, events and methods.

The **wj-multi-select** component may contain a **WjItemTemplate** child directive.

Constructor

- constructor

Properties

- asyncBindings
- autoExpandSelection
- checkedItems
- checkedItemsChangedNg
- checkedMemberPath
- collectionView
- controlTemplate
- displayMemberPath
- dropDown
- dropDownCssClass
- formatItem
- formatItemNg
- gotFocusNg
- headerFormat
- headerFormatter
- headerPath
- hostElement
- initialized
- inputElement
- isAnimated
- isContentHtml
- isDisabled
- isDroppedDown
- isDroppedDownChangedNg
- isDroppedDownChangingNg
- isEditable
- isInitialized
- isReadOnly
- isRequired
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- listBox
- lostFocusNg
- maxDropDownHeight
- maxDropDownWidth
- maxHeaderItems
- placeholder
- rightToLeft
- selectAllLabel
- selectedIndex
- selectedIndexChangedNg
- selectedItem
- selectedValue
- selectedValuePath
- showDropDownButton
- showSelectAllCheckbox
- text
- textChangedNg
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getDisplayText
- getTemplate
- indexOf
- initialize
- invalidate
- invalidateAll
- onCheckedItemsChanged
- onGotFocus
- onIsDroppedDownChanged
- onIsDroppedDownChanging
- onItemsSourceChanged
- onLostFocus
- onSelectedIndexChanged
- onTextChanged
- refresh
- refreshAll
- removeEventListener
- selectAll

Events

- checkedItemsChanged
- gotFocus
- isDroppedDownChanged
- isDroppedDownChanging
- itemsSourceChanged
- lostFocus
- selectedIndexChanged
- textChanged

Constructor

constructor

```
constructor(element: any, options?): MultiSelect
```

Initializes a new instance of the **MultiSelect** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
MultiSelect
Returns
MultiSelect

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

autoExpandSelection

Gets or sets a value that indicates whether the control should automatically expand the selection to whole words/numbers when the control is clicked.

Inherited From
`DropDown`
Type
boolean

checkedItems

Gets or sets an array containing the items that are currently checked.

Inherited From
`MultiSelect`
Type
any[]

checkedItemsChangedNg

Angular (EventEmitter) version of the Wijmo **checkedItemsChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **checkedItemsChanged** Wijmo event name.

Type
EventEmitter

checkedMemberPath

Gets or sets the name of the property used to control the checkboxes placed next to each item.

Inherited From
`MultiSelect`
Type
string

collectionView

Gets the **`ICollectionView`** object used as the item source.

Inherited From
`ComboBox`
Type
`ICollectionView`

STATIC controlTemplate

Gets or sets the template used to instantiate **`DropDown`** controls.

Inherited From
`DropDown`
Type
any

displayMemberPath

Gets or sets the name of the property to use as the visual representation of the items.

Inherited From
`ComboBox`
Type
string

dropDown

Gets the drop down element shown when the **`isDroppedDown`** property is set to true.

Inherited From
`DropDown`
Type
HTMLElement

dropDownCssClass

Gets or sets a CSS class name to add to the control's drop-down element.

This property is useful when styling the drop-down element, because it is shown as a child of the document body rather than as a child of the control itself, which prevents using CSS selectors based on the parent control.

Inherited From
`DropDown`
Type
string

formatItem

Event that fires when items in the drop-down list are created.

You can use this event to modify the HTML in the list items. For details, see the **`formatItem`** event.

Inherited From
`ComboBox`
Type
`Event`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

headerFormat

Gets or sets the format string used to create the header content when the control has more than **`maxHeaderItems`** items checked.

The format string may contain the '{count}' replacement string which gets replaced with the number of items currently checked. The default value for this property in the English culture is '{count:n0} items selected'.

Inherited From
`MultiSelect`
Type
string

headerFormatter

Gets or sets a function that gets the HTML in the control header.

By default, the control header content is determined based on the **`placeholder`**, **`maxHeaderItems`**, and on the current selection.

You may customize the header content by specifying a function that returns a custom string based on whatever criteria your application requires.

Inherited From
`MultiSelect`
Type
Function

headerPath

Gets or sets the name of a property to use for getting the value displayed in the control's input element.

The default value for this property is null, which causes the control to display the same content in the input element as in the selected item of the drop-down list.

Use this property if you want to de-couple the value shown in the input element from the values shown in the drop-down list. For example, the input element could show an item's name and the drop-down list could show additional detail.

Inherited From
`ComboBox`
Type
string

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputElement

Gets the HTML input element hosted by the control.

Use this property in situations where you want to customize the attributes of the input element.

Inherited From
`DropDown`
Type
HTMLInputElement

isAnimated

Gets or sets a value that indicates whether the control should use a fade-in animation when displaying the drop-down.

Inherited From
`DropDown`
Type
boolean

isContentHtml

Gets or sets a value indicating whether the drop-down list displays items as plain text or as HTML.

Inherited From
`ComboBox`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isDroppedDown

Gets or sets a value that indicates whether the drop down is currently visible.

Inherited From
`DropDown`
Type
boolean

isDroppedDownChangedNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanged** Wijmo event name.

Type
EventEmitter

isDroppedDownChangingNg

Angular (EventEmitter) version of the Wijmo **isDroppedDownChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **isDroppedDownChanging** Wijmo event name.

Type
EventEmitter

isEditable

Gets or sets a value that determines whether the content of the input element should be restricted to items in the **`itemsSource`** collection.

Inherited From
`ComboBox`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether the user can modify the control value using the mouse and keyboard.

Inherited From
`DropDown`
Type
boolean

isRequired

Gets or sets a value that determines whether the control value must be set to a non-null value or whether it can be set to null (by deleting the content of the control).

Inherited From
`DropDown`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a function used to customize the values shown in the drop-down list. The function takes two arguments, the item index and the default text or html, and returns the new text or html to display.

If the formatting function needs a scope (i.e. a meaningful 'this' value), then remember to set the filter using the 'bind' function to specify the 'this' object. For example:

```
comboBox.itemFormatter = customItemFormatter.bind(this);
  function customItemFormatter(index, content) {
    if (this.makeItemBold(index)) {
      content = '<b>' + content + '</b>';
    }
    return content;
  }
```

Inherited From
`ComboBox`
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the items to select from.

Inherited From
ComboBox
Type
any

listBox

Gets the **ListBox** control shown in the drop-down.

Inherited From
ComboBox
Type
ListBox

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxDropDownHeight

Gets or sets the maximum height of the drop-down list.

Inherited From
ComboBox
Type
number

maxDropDownWidth

Gets or sets the maximum width of the drop-down list.

The width of the drop-down list is also limited by the width of the control itself (that value represents the drop-down's minimum width).

Inherited From
`ComboBox`
Type
number

maxHeaderItems

Gets or sets the maximum number of items to display on the control header.

If no items are selected, the header displays the text specified by the **`placeholder`** property.

If the number of selected items is smaller than or equal to the value of the **`maxHeaderItems`** property, the selected items are shown in the header.

If the number of selected items is greater than **`maxHeaderItems`**, the header displays the selected item count instead.

Inherited From
`MultiSelect`
Type
number

placeholder

Gets or sets the string shown as a hint when the control is empty.

Inherited From
`DropDown`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectAllLabel

Gets or sets the string to be used as a label for the "Select All" checkbox that is displayed when the **showSelectAllCheckbox** property is set to true.

This property is set to null by default, which causes the control to show a localized version of the string "Select All".

Inherited From
`MultiSelect`
Type
string

selectedIndex

Gets or sets the index of the currently selected item in the drop-down list.

Inherited From
`ComboBox`
Type
number

selectedIndexChangedNg

Angular (EventEmitter) version of the Wijmo **selectedIndexChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedIndexChanged** Wijmo event name.

Type
EventEmitter

selectedItem

Gets or sets the item that is currently selected in the drop-down list.

Inherited From
`ComboBox`
Type
any

selectedValue

Gets or sets the value of the **`selectedItem`**, obtained using the **`selectedValuePath`**.

Inherited From
`ComboBox`
Type
any

selectedValuePath

Gets or sets the name of the property used to get the **`selectedValue`** from the **`selectedItem`**.

Inherited From
`ComboBox`
Type
string

showDropDownButton

Gets or sets a value that indicates whether the control should display a drop-down button.

Inherited From
`DropDown`
Type
boolean

showSelectAllCheckbox

Gets or sets whether the control should display a "Select All" checkbox above the items to select or de-select all items.

Inherited From
`MultiSelect`
Type
boolean

text

Gets or sets the text shown on the control.

Inherited From
`DropDown`
Type
string

textChangedNg

Angular (EventEmitter) version of the Wijmo **textChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **textChanged** Wijmo event name.

Type
EventEmitter

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is 'checkedItems'.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getDisplayText

```
getDisplayText(index?: number): string
```

Gets the string displayed in the input element for the item at a given index (always plain text).

Parameters

- **index**: **number** OPTIONAL
  The index of the item to retrieve the text for.

Inherited From
`ComboBox`
Returns
string

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

indexOf

```
indexOf(text: string, fullMatch: boolean): number
```

Gets the index of the first item that matches a given string.

Parameters

- **text**: **string**
  The text to search for.
- **fullMatch**: **boolean**
  Whether to look for a full match or just the start of the string.

Inherited From
ComboBox
Returns
number

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onCheckedItemsChanged

```
onCheckedItemsChanged(e?: EventArgs): void
```

Raises the **checkedItemsChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`MultiSelect`
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`Control`
Returns
void

onIsDroppedDownChanged

```
onIsDroppedDownChanged(e?: EventArgs): void
```

Raises the **isDroppedDownChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
`DropDown`
Returns
void

onIsDroppedDownChanging

```
onIsDroppedDownChanging(e: CancelEventArgs): boolean
```

Raises the **isDroppedDownChanging** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
DropDown
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onSelectedIndexChanged

```
onSelectedIndexChanged(e?: EventArgs): void
```

Raises the **selectedIndexChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
ComboBox
Returns
void

onTextChanged

```
onTextChanged(e?: EventArgs): void
```

Raises the **textChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
DropDown
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

selectAll

```
selectAll(): void
```

Sets the focus to the control and selects all its content.

Inherited From
`DropDown`
Returns
void

Events

checkedItemsChanged

Occurs when the value of the **`checkedItems`** property changes.

Inherited From
`MultiSelect`
Arguments
`EventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

isDroppedDownChanged

Occurs after the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`EventArgs`

isDroppedDownChanging

Occurs before the drop down is shown or hidden.

Inherited From
`DropDown`
Arguments
`CancelEventArgs`

itemsSourceChanged

Occurs when the value of the **`itemsSource`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

selectedIndexChanged

Occurs when the value of the **`selectedIndex`** property changes.

Inherited From
`ComboBox`
Arguments
`EventArgs`

textChanged

Occurs when the value of the **`text`** property changes.

Inherited From
`DropDown`
Arguments
`EventArgs`

WjPopup Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.input
Base Class
Popup

Angular 2 component for the **Popup** control.

Use the **wj-popup** component to add **Popup** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjPopup** component is derived from the **Popup** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- content
- dialogResult
- dialogResultEnter
- fadeIn
- fadeOut
- gotFocusNg
- hiddenNg
- hideTrigger
- hidingNg
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- isVisible
- lostFocusNg
- modal
- owner
- removeOnHide
- rightToLeft
- showingNg
- shownNg
- showTrigger
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hide
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onHidden
- onHiding
- onLostFocus
- onShowing
- onShown
- refresh
- refreshAll
- removeEventListener
- show

Events

- gotFocus
- hidden
- hiding
- lostFocus
- showing
- shown

Constructor

constructor

```
constructor(element: any, options?: any): Popup
```

Initializes a new instance of the **Popup** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the control.

Inherited From
Popup
Returns
Popup

Properties

content

Gets or sets the HTML element contained in this **Popup**.

Inherited From
Popup
Type
HTMLElement

dialogResult

Gets or sets a value that can be used for handling the content of the **Popup** after it is hidden.

This property is set to null when the **Popup** is displayed, and it can be set in response to button click events or in the call to the **hide** method.

Inherited From
Popup
Type
any

dialogResultEnter

Gets or sets a value to be used as a **dialogResult** when the user presses the Enter key while the **Popup** is visible.

If the user presses Enter and the **dialogResultEnter** property is not null, the popup checks whether all its child elements are in a valid state. If so, the popup is closed and the **dialogResult** property is set to the value of the **dialogResultEnter** property.

Inherited From
Popup
Type
any

fadeIn

Gets or sets a value that determines whether the **Popup** should use a fade-out animation when it is shown.

Inherited From
Popup
Type
boolean

fadeOut

Gets or sets a value that determines whether the **Popup** should use a fade-out animation when it is hidden.

Inherited From
Popup
Type
boolean

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

hiddenNg

Angular (EventEmitter) version of the Wijmo **hidden** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **hidden** Wijmo event name.

Type
EventEmitter

hideTrigger

Gets or sets the actions that hide the **Popup**.

By default, the **hideTrigger** property is set to **Blur**, which hides the popup when it loses focus.

If you set the **hideTrigger** property to **Click**, the popup will be hidden only when the owner element is clicked.

If you set the **hideTrigger** property to **None**, the popup will be hidden only when the **hide** method is called.

Inherited From
Popup
Type
PopupTrigger

hidingNg

Angular (EventEmitter) version of the Wijmo **hiding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **hiding** Wijmo event name.

Type
EventEmitter

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

isVisible

Gets a value that determines whether the **Popup** is currently visible.

Inherited From
Popup
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

modal

Gets or sets a value that determines whether the **Popup** should be displayed as a modal dialog.

Modal dialogs show a dark backdrop that makes the **Popup** stand out from other content on the page.

If you want to make a dialog truly modal, also set the **hideTrigger** property to **None**, so users won't be able to click the backdrop to dismiss the dialog. In this case, the dialog will close only if the **hide** method is called or if the user presses the Escape key.

Inherited From
Popup
Type
boolean

owner

Gets or sets the element that owns this **Popup**.

If the **owner** is null, the **Popup** behaves like a dialog. It is centered on the screen and must be shown using the **show** method.

Inherited From
Popup
Type
HTMLElement

removeOnHide

Gets or sets a value that determines whether the **Popup** element should be removed from the DOM when the **Popup** is hidden, as opposed to being hidden.

This property is set to true by default.

Inherited From
`Popup`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

showingNg

Angular (EventEmitter) version of the Wijmo **showing** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **showing** Wijmo event name.

Type
EventEmitter

shownNg

Angular (EventEmitter) version of the Wijmo **shown** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **shown** Wijmo event name.

Type
EventEmitter

showTrigger

Gets or sets the actions that show the **Popup**.

By default, the **showTrigger** property is set to **Click**, which causes the popup to appear when the user clicks the owner element.

If you set the **showTrigger** property to **None**, the popup will be shown only when the **show** method is called.

Inherited From
Popup
Type
PopupTrigger

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hide

```
hide(dialogResult?: any): void
```

Hides the **Popup**.

Parameters

- **dialogResult**: **any** OPTIONAL
  Optional value assigned to the **dialogResult** property before closing the **Popup**.

Inherited From
Popup
Returns
void

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onHidden

```
onHidden(e?: EventArgs): void
```

Raises the **hidden** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Popup
Returns
void

onHiding

```
onHiding(e: CancelEventArgs): boolean
```

Raises the **hiding** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
Popup
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onShowing

```
onShowing(e: CancelEventArgs): boolean
```

Raises the **showing** event.

Parameters

- **e**: **CancelEventArgs**

Inherited From
Popup
Returns
boolean

onShown

```
onShown(e?: EventArgs): void
```

Raises the **shown** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Popup
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the control.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

show

```
show(modal?: boolean, handleResult?: Function): void
```

Shows the **Popup**.

Parameters

- **modal**: **boolean** OPTIONAL
  Whether to show the popup as a modal dialog. If provided, this sets the value of the **modal** property.

- **handleResult**: **Function** OPTIONAL
  Callback invoked when the popup is hidden. If provided, this should be a function that receives the popup as a parameter.

  The **handleResult** callback allows callers to handle the result of modal dialogs without attaching handlers to the **hidden** event. For example, the code below shows a dialog used to edit the current item in a **CollectionView**. The edits are committed or canceled depending on the **dialogResult** value. For example:

```
$scope.editCurrentItem = function () {
  $scope.data.editItem($scope.data.currentItem);
  $scope.itemEditor.show(true, function (e) {
    if (e.dialogResult == 'wj-hide-ok') {
      $scope.data.commitEdit();
    } else {
      $scope.data.cancelEdit();
    }
  });
}
```

Inherited From
Popup
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
Control
Arguments
EventArgs

hidden

Occurs after the **Popup** has been hidden.

Inherited From
Popup
Arguments
EventArgs

hiding

Occurs before the **Popup** is hidden.

Inherited From
Popup
Arguments
CancelEventArgs

lostFocus

Occurs when the control loses the focus.

Inherited From
Control
Arguments
EventArgs

showing

Occurs before the **Popup** is shown.

Inherited From
Popup
Arguments
CancelEventArgs

shown

Occurs after the **Popup** has been shown.

Inherited From
Popup
Arguments
EventArgs

wijmo/wijmo.angular2.grid Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid

Contains Angular 2 components for the **wijmo.grid** module.

wijmo.angular2.grid is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
<p>Here is a data bound FlexGrid control with four columns:</p>
<wj-flex-grid [itemsSource]="data">
  <wj-flex-grid-column
    [header]="'Country'"
    [binding]="'country'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Sales'"
    [binding]="'sales'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Expenses'"
    [binding]="'expenses'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Downloads'"
    [binding]="'downloads'">
  </wj-flex-grid-column>
</wj-flex-grid>
```

Classes

- WjFlexGrid
- WjFlexGridCellTemplate
- WjFlexGridColumn

Enums

- CellTemplateType

WjFlexGrid Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid
Base Class
FlexGrid

Angular 2 component for the **FlexGrid** control.

Use the **wj-flex-grid** component to add **FlexGrid** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup. For example:

```
<p>Here is a data bound FlexGrid control with four columns:</p>
<wj-flex-grid [itemsSource]="data">
  <wj-flex-grid-column
    [header]="'Country'"
    [binding]="'country'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Sales'"
    [binding]="'sales'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Expenses'"
    [binding]="'expenses'">
  </wj-flex-grid-column>
  <wj-flex-grid-column
    [header]="'Downloads'"
    [binding]="'downloads'">
  </wj-flex-grid-column>
</wj-flex-grid>
```

The **WjFlexGrid** component is derived from the **FlexGrid** control and inherits all its properties, events and methods. The following properties are not available for binding in templates: **scrollPosition**, **selection** and **columnLayout** properties.

The **wj-flex-grid** component may contain the following child components: **WjFlexGridDetail** , **WjFlexGridFilter** , **WjFlexGridColumn** and **WjFlexGridCellTemplate**.

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells
- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- copiedNg
- copyingNg
- deferResizing
- deletedRowNg
- deletingRowNg
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowNg
- draggingRowOverNg
- editableCollectionView
- editRange
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab
- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager
- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft
- rowAddedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg
- rowEditStartingNg
- rowHeaderPath
- rowHeaders

- rows
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded

- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): FlexGrid
```

Initializes a new instance of the **FlexGrid** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Inherited From
FlexGrid
Returns
FlexGrid

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Inherited From
FlexGrid
Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **isReadOnly** property is set to true.

Inherited From
FlexGrid
Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Inherited From
`FlexGrid`
Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Inherited From
`FlexGrid`
Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Inherited From
`FlexGrid`
Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Inherited From
`FlexGrid`
Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Inherited From
`FlexGrid`
Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **itemsSource**.

The column generation depends on the **itemsSource** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **autoGenerateColumns** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Inherited From
FlexGrid
Type
boolean

autoSizedColumnNg

Angular (EventEmitter) version of the Wijmo **autoSizedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizedColumn** Wijmo event name.

Type
EventEmitter

autoSizedRowNg

Angular (EventEmitter) version of the Wijmo **autoSizedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizedRow** Wijmo event name.

Type
EventEmitter

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Inherited From
`FlexGrid`
Type
`AutoSizeMode`

autoSizingColumnNg

Angular (EventEmitter) version of the Wijmo **autoSizingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizingColumn** Wijmo event name.

Type
EventEmitter

autoSizingRowNg

Angular (EventEmitter) version of the Wijmo **autoSizingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizingRow** Wijmo event name.

Type
EventEmitter

beginningEditNg

Angular (EventEmitter) version of the Wijmo **beginningEdit** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **beginningEdit** Wijmo event name.

Type
EventEmitter

bottomLeftCells

Gets the **GridPanel** that contains the bottom left cells.

The **bottomLeftCells** panel appears below the row headers, to the left of the **columnFooters** panel.

Inherited From
FlexGrid
Type
GridPanel

cellEditEndedNg

Angular (EventEmitter) version of the Wijmo **cellEditEnded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **cellEditEnded** Wijmo event name.

Type
EventEmitter

cellEditEndingNg

Angular (EventEmitter) version of the Wijmo **cellEditEnding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **cellEditEnding** Wijmo event name.

Type
EventEmitter

cellFactory

Gets or sets the **CellFactory** that creates and updates cells for this grid.

Inherited From
FlexGrid
Type
CellFactory

cells

Gets the **GridPanel** that contains the data cells.

Inherited From
FlexGrid
Type
GridPanel

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Inherited From
FlexGrid
Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Inherited From
FlexGrid
Type
Size

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Inherited From
`FlexGrid`
Type
boolean

collectionView

Gets the **`ICollectionView`** that contains the grid data.

Inherited From
`FlexGrid`
Type
`ICollectionView`

columnFooters

Gets the **`GridPanel`** that contains the column footer cells.

The **`columnFooters`** panel appears below the grid cells, to the right of the **`bottomLeftCells`** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **`columnFooters`** panel to display summary data for columns that have the **`aggregate`** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Inherited From
`FlexGrid`
Type
`GridPanel`

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Inherited From
FlexGrid
Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Inherited From
FlexGrid
Type
string

columns

Gets the grid's column collection.

Inherited From
FlexGrid
Type
ColumnCollection

controlRect

Gets the bounding rectangle of the control in page coordinates.

Inherited From
FlexGrid
Type
Rect

STATIC controlTemplate

Gets or sets the template used to instantiate **`FlexGrid`** controls.

Inherited From
`FlexGrid`
Type
any

copiedNg

Angular (EventEmitter) version of the Wijmo **copied** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **copied** Wijmo event name.

Type
EventEmitter

copyingNg

Angular (EventEmitter) version of the Wijmo **copying** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **copying** Wijmo event name.

Type
EventEmitter

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **`deferResizing`** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Inherited From
`FlexGrid`
Type
boolean

deletedRowNg

Angular (EventEmitter) version of the Wijmo **deletedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **deletedRow** Wijmo event name.

Type
EventEmitter

deletingRowNg

Angular (EventEmitter) version of the Wijmo **deletingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **deletingRow** Wijmo event name.

Type
EventEmitter

draggedColumnNg

Angular (EventEmitter) version of the Wijmo **draggedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggedColumn** Wijmo event name.

Type
EventEmitter

draggedRowNg

Angular (EventEmitter) version of the Wijmo **draggedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggedRow** Wijmo event name.

Type
EventEmitter

draggingColumnNg

Angular (EventEmitter) version of the Wijmo **draggingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingColumn** Wijmo event name.

Type
EventEmitter

draggingColumnOverNg

Angular (EventEmitter) version of the Wijmo **draggingColumnOver** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingColumnOver** Wijmo event name.

Type
EventEmitter

draggingRowNg

Angular (EventEmitter) version of the Wijmo **draggingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRow** Wijmo event name.

Type
EventEmitter

draggingRowOverNg

Angular (EventEmitter) version of the Wijmo **draggingRowOver** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRowOver** Wijmo event name.

Type
EventEmitter

editableCollectionView

Gets the **`IEditableCollectionView`** that contains the grid data.

Inherited From
`FlexGrid`
Type
`IEditableCollectionView`

editRange

Gets a **`CellRange`** that identifies the cell currently being edited.

Inherited From
`FlexGrid`
Type
`CellRange`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

groupCollapsedChangedNg

Angular (EventEmitter) version of the Wijmo **groupCollapsedChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **groupCollapsedChanged** Wijmo event name.

Type
EventEmitter

groupCollapsedChangingNg

Angular (EventEmitter) version of the Wijmo **groupCollapsedChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **groupCollapsedChanging** Wijmo event name.

Type
EventEmitter

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Inherited From
FlexGrid
Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Inherited From
`FlexGrid`
Type
boolean

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Inherited From
`FlexGrid`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Inherited From
FlexGrid
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** that contains items shown on the grid.

Inherited From
`FlexGrid`
Type
any

itemsSourceChangedNg

Angular (EventEmitter) version of the Wijmo **itemsSourceChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **itemsSourceChanged** Wijmo event name.

Type
EventEmitter

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **`getError`** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Inherited From
`FlexGrid`
Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **`MoveDown`**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Inherited From
`FlexGrid`
Type
`KeyAction`

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **`None`**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **`Cycle`**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **`CycleOut`** setting that causes the selection to move through the cells (as **`Cycle`**), and then on to the next/previous control on the page when the last or first cells are selected.

Inherited From
`FlexGrid`
Type
`KeyAction`

loadedRowsNg

Angular (EventEmitter) version of the Wijmo **loadedRows** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loadedRows** Wijmo event name.

Type
EventEmitter

loadingRowsNg

Angular (EventEmitter) version of the Wijmo **loadingRows** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loadingRows** Wijmo event name.

Type
EventEmitter

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mergeManager

Gets or sets the **`MergeManager`** object responsible for determining how cells should be merged.

Inherited From
`FlexGrid`
Type
`MergeManager`

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **`newRowAtTop`** property to true, and you want the new row template to remain visible at all times, set the **`frozenRows`** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **`allowAddNew`** property is set to true and if the **`itemsSource`** object supports adding new items.

Inherited From
`FlexGrid`
Type
boolean

pastedCellNg

Angular (EventEmitter) version of the Wijmo **pastedCell** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pastedCell** Wijmo event name.

Type
EventEmitter

pastedNg

Angular (EventEmitter) version of the Wijmo **pasted** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pasted** Wijmo event name.

Type
EventEmitter

pastingCellNg

Angular (EventEmitter) version of the Wijmo **pastingCell** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pastingCell** Wijmo event name.

Type
EventEmitter

pastingNg

Angular (EventEmitter) version of the Wijmo **pasting** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pasting** Wijmo event name.

Type
EventEmitter

prepareCellForEditNg

Angular (EventEmitter) version of the Wijmo **prepareCellForEdit** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **prepareCellForEdit** Wijmo event name.

Type
EventEmitter

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **`preserveOutlineState`** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Inherited From
`FlexGrid`
Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Inherited From
`FlexGrid`
Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Inherited From
`FlexGrid`
Type
boolean

resizedColumnNg

Angular (EventEmitter) version of the Wijmo **resizedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizedColumn** Wijmo event name.

Type
EventEmitter

resizedRowNg

Angular (EventEmitter) version of the Wijmo **resizedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizedRow** Wijmo event name.

Type
EventEmitter

resizingColumnNg

Angular (EventEmitter) version of the Wijmo **resizingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizingColumn** Wijmo event name.

Type
EventEmitter

resizingRowNg

Angular (EventEmitter) version of the Wijmo **resizingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizingRow** Wijmo event name.

Type
EventEmitter

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowAddedNg

Angular (EventEmitter) version of the Wijmo **rowAdded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowAdded** Wijmo event name.

Type
EventEmitter

rowEditEndedNg

Angular (EventEmitter) version of the Wijmo **rowEditEnded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditEnded** Wijmo event name.

Type
EventEmitter

rowEditEndingNg

Angular (EventEmitter) version of the Wijmo **rowEditEnding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditEnding** Wijmo event name.

Type
EventEmitter

rowEditStartedNg

Angular (EventEmitter) version of the Wijmo **rowEditStarted** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditStarted** Wijmo event name.

Type
EventEmitter

rowEditStartingNg

Angular (EventEmitter) version of the Wijmo **rowEditStarting** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditStarting** Wijmo event name.

Type
EventEmitter

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Inherited From
`FlexGrid`
Type
string

rowHeaders

Gets the `GridPanel` that contains the row header cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

rows

Gets the grid's row collection.

Inherited From
`FlexGrid`
Type
`RowCollection`

scrollPosition

Gets or sets a `Point` that represents the value of the grid's scrollbars.

Inherited From
`FlexGrid`
Type
`Point`

scrollPositionChangedNg

Angular (EventEmitter) version of the Wijmo **scrollPositionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **scrollPositionChanged** Wijmo event name.

Type
EventEmitter

scrollSize

Gets the size of the grid content in pixels.

Inherited From
`FlexGrid`
Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selection

Gets or sets the current selection.

Inherited From
`FlexGrid`
Type
`CellRange`

selectionChangedNg

Angular (EventEmitter) version of the Wijmo **selectionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanged** Wijmo event name.

Type
EventEmitter

selectionChangingNg

Angular (EventEmitter) version of the Wijmo **selectionChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanging** Wijmo event name.

Type
EventEmitter

selectionMode

Gets or sets the current selection mode.

Inherited From
`FlexGrid`
Type
`SelectionMode`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Inherited From
`FlexGrid`
Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **`showDropDown`** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`FlexGrid`
Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **`itemValidator`** property or the **`getError`** property on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Inherited From
`FlexGrid`
Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

sortedColumnNg

Angular (EventEmitter) version of the Wijmo **sortedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sortedColumn** Wijmo event name.

Type
EventEmitter

sortingColumnNg

Angular (EventEmitter) version of the Wijmo **sortingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sortingColumn** Wijmo event name.

Type
EventEmitter

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Inherited From
`FlexGrid`
Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Inherited From
`FlexGrid`
Type
boolean

topLeftCells

Gets the **GridPanel** that contains the top left cells (to the left of the column headers).

Inherited From
FlexGrid
Type
GridPanel

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Inherited From
FlexGrid
Type
number

updatedLayoutNg

Angular (EventEmitter) version of the Wijmo **updatedLayout** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatedLayout** Wijmo event name.

Type
EventEmitter

updatedViewNg

Angular (EventEmitter) version of the Wijmo **updatedView** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatedView** Wijmo event name.

Type
EventEmitter

updatingLayoutNg

Angular (EventEmitter) version of the Wijmo **updatingLayout** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatingLayout** Wijmo event name.

Type
EventEmitter

updatingViewNg

Angular (EventEmitter) version of the Wijmo **updatingView** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatingView** Wijmo event name.

Type
EventEmitter

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **`getError`** method on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

viewRange

Gets the range of cells currently in view.

Inherited From
`FlexGrid`
Type
`CellRange`

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Inherited From
`FlexGrid`
Type
number

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.

- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`Control`
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
FlexGrid
Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Inherited From
FlexGrid
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
`Control`
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Inherited From
`FlexGrid`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
`Control`
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **`beginUpdate`**.

Inherited From
`Control`
Returns
void

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Inherited From
FlexGrid
Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Inherited From
FlexGrid
Returns
void

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
`FlexGrid`
Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Inherited From
FlexGrid
Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Inherited From
`FlexGrid`
Returns
`Column`

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Inherited From
FlexGrid
Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.
- **c**: **number**
  Column index of the cell to inspect.

Inherited From
FlexGrid
Returns
SelectedState

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Inherited From
FlexGrid
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Inherited From
FlexGrid
Returns
boolean

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the grid layout and content, or just the content.

Inherited From
FlexGrid
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.

- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.

- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Inherited From
FlexGrid
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.
- **c**: **number**
  Index of the column to scroll into view.

Inherited From
FlexGrid
Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

Parameters

- **rng**: **any**
  Range to select.
- **show**: **any** OPTIONAL
  Whether to scroll the new selection into view.

Inherited From
FlexGrid
Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Sets the value of a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the grid to show the change.

Inherited From
`FlexGrid`
Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Hidden rows and columns are skipped.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Inherited From
FlexGrid
Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Inherited From
`FlexGrid`
Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Inherited From
`FlexGrid`
Arguments
`CellEditEndingEventArgs`

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **allowDelete** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **allowDelete** property).

The event handler may cancel the row deletion.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedColumn

Occurs when the user finishes dragging a column.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedRow

Occurs when the user finishes dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRowOver

Occurs as the user drags a row to a new position.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **`itemFormatter`** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Inherited From
`FlexGrid`
Arguments
`FormatItemEventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

resizedColumn

Occurs when the user finishes resizing a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizedRow

Occurs when the user finishes resizing rows.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **`rowEditStarted`** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarted

Occurs after a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

selectionChanged

Occurs after selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

updatedLayout

Occurs after the grid has updated its internal layout.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatingLayout

Occurs before the grid updates its internal layout.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

WjFlexGridCellTemplate Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid

Angular 2 directive for the **FlexGrid** cell templates.

The **wjFlexGridCellTemplate** directive defines a template for a certain cell type in **FlexGrid**. The template should be defined on a **<template>** element and must contain a **cellType** attribute that specifies the **CellTemplateType**. Depending on the template's cell type, the **<template>** element with the **wjFlexGridCellTemplate** directive must be a child of either **WjFlexGrid** or **WjFlexGridColumn** directives.

Column-specific cell templates must be contained in **wj-flex-grid-column** components, and cells that are not column-specific (like row header or top left cells) must be contained in the **wj-flex-grid** component.

The **<template>** element with the **wjFlexGridCellTemplate** directive may contain an arbitrary HTML fragment with Angular 2 interpolation expressions and other components and directives.

Bindings in HTML fragment can use the **cell** local template variable containing the cell context object, with **col**, **row**, and **item** properties that refer to the **Column**, **Row**, and **Row.dataItem** objects pertaining to the cell.

For cell types like **Group** and **CellEdit**, an additional **value** property containing an unformatted cell value is provided. For example, here is a **FlexGrid** control with templates for row header cells and, regular and column header cells of the Country column:

```
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjGrid.WjFlexGridColumn, wjGrid.WjFlexGridCellTemplate],
    template: `
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeader'" let-cell="cell">
    {{cell.row.index}}
  </template>
  <template wjFlexGridCellTemplate [cellType]="'RowHeaderEdit'">
    ...
  </template>

  <wj-flex-grid-column [header]="'Country'" [binding]="'country'">
    <template wjFlexGridCellTemplate [cellType]="'ColumnHeader'" let-cell="cell">
      <img src="resources/globe.png" />
        {{cell.col.header}}
    </template>
    <template wjFlexGridCellTemplate [cellType]="'Cell'" let-cell="cell">
      <img src="resources/{{cell.item.country}}.png" />
      {{cell.item.country}}
    </template>
  </wj-flex-grid-column>
  <wj-flex-grid-column [header]="'Sales'" [binding]="'sales'"></wj-flex-grid-column>
</wj-flex-grid>
`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

For more detailed information on specific cell type templates, refer to the documentation for **CellTemplateType** enumeration.

The **wjFlexGridCellTemplate** directive supports the following attributes:

cellType
The **CellTemplateType** value defining the type of cell to which the template is applied.
cellOverflow
Defines the **style.overflow** property value for cells.

The **cellType** attribute takes any of the following enumerated values:

Cell

Defines a regular (data) cell template. Must be a child of the **WjFlexGridColumn** component. For example, this cell template shows flags in the cells of Country column:

```
<wj-flex-grid-column [header]="'Country'" [binding]="'country'">
  <template wjFlexGridCellTemplate [cellType]="'Cell'" let-cell="cell">
    <img src="resources/{{cell.item.country}}.png" />
    {{cell.item.country}}
  </template>
</wj-flex-grid-column>
```

If **Group** template is not provided for a hierarchical **FlexGrid** (that is, one with the **childItemsPath** property specified), non-header cells in group rows of this **Column** also use this template.

CellEdit

Defines a template for a cell in edit mode. Must be a child of the **WjFlexGridColumn** component. This cell type has an additional **cell.value** property available for binding. It contains the original cell value before editing, and the updated value after editing. For example, here is a template that uses the Wijmo **InputNumber** control as an editor for the "Sales" column:

```
<wj-flex-grid-column [header]="'Sales'" [binding]="'sales'">
  <template wjFlexGridCellTemplate [cellType]="'CellEdit'">
    <wj-input-number [(value)]="cell.value" [step]="1"></wj-input-number>
  </template>
</wj-flex-grid-column>
```

ColumnHeader

Defines a template for a column header cell. Must be a child of the **WjFlexGridColumn** component. For example, this template adds an image to the header of the "Country" column:

```
<wj-flex-grid-column [header]="'Country'" [binding]="'country'">
  <template wjFlexGridCellTemplate [cellType]="'ColumnHeader'" let-cell="cell">
    <img src="resources/globe.png" />
      {{cell.col.header}}
  </template>
</wj-flex-grid-column>
```

RowHeader

Defines a template for a row header cell. Must be a child of the **WjFlexGrid** component. For example, this template shows row indices in the row headers:

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeader'" let-cell="cell">
    {{cell.row.index + 1}}
  </template>
</wj-flex-grid>
```

Note that this template is applied to a row header cell, even if it is in a row that is in edit mode. In order to provide an edit-mode version of a row header cell with alternate content, define the **RowHeaderEdit** template.

RowHeaderEdit

Defines a template for a row header cell in edit mode. Must be a child of the **WjFlexGrid** component. For example, this template shows dots in the header of rows being edited:

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeaderEdit'">
    ...
  </template>
</wj-flex-grid>
```

Use the following **RowHeaderEdit** template to add the standard edit-mode indicator to cells where the **RowHeader** template applies:

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'RowHeaderEdit'">
    {{&#x270e;}}
  </template>
</wj-flex-grid>
```

TopLeft

Defines a template for the top left cell. Must be a child of the **WjFlexGrid** component. For example, this template shows a down/right glyph in the top-left cell of the grid:

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'TopLeft'">
    <span class="wj-glyph-down-right"></span>
  </template>
</wj-flex-grid>
```

GroupHeader

Defines a template for a group header cell in a **GroupRow**, Must be a child of the **WjFlexGridColumn** component.

The **cell.row** property contains an instance of the **GroupRow** class. If the grouping comes from **CollectionView**, the **cell.item** property references the **CollectionViewGroup** object.

For example, this template uses a checkbox element as an expand/collapse toggle:

```
<wj-flex-grid-column [header]="'Country'" [binding]="'country'">
  <template wjFlexGridCellTemplate [cellType]="'GroupHeader'" let-cell="cell">
    <input type="checkbox" [(ngModel)]="cell.row.isCollapsed"/>
    {{cell.item.name}} ({{cell.item.items.length}} items)
  </template>
</wj-flex-grid-column>
```

Group

Defines a template for a regular cell (not a group header) in a **GroupRow**. Must be a child of the **WjFlexGridColumn** component. This cell type has an additional **cell.value** property available for binding. In cases where columns have the **aggregate** property specified, it contains the unformatted aggregate value.

For example, this template shows aggregate's value and kind for group row cells in the "Sales" column:

```
<wj-flex-grid-column [header]="'Sales'" [binding]="'sales'" [aggregate]="'Avg'">
  <template wjFlexGridCellTemplate [cellType]="'Group'" let-cell="cell">
    Average: {{cell.value | number:'1.0-0'}}
  </template>
</wj-flex-grid-column>
```

ColumnFooter

Defines a template for a regular cell in a **columnFooters** panel. Must be a child of the **WjFlexGridColumn** component. This cell type has an additional **cell.value** property available for binding that contains a cell value.

For example, this template shows aggregate's value and kind for a footer cell in the "Sales" column:

```
<wj-flex-grid-column [header]="'Sales'" [binding]="'sales'" [aggregate]="'Avg'">
  <template wjFlexGridCellTemplate [cellType]="'ColumnFooter'" let-cell="cell">
    Average: {{cell.value | number:'1.0-0'}}
  </template>
</wj-flex-grid-column>
```

BottomLeft

Defines a template for the bottom left cells (at the intersection of the row header and column footer cells). Must be a child of the **WjFlexGrid** component. For example, this template shows a sigma glyph in the bottom-left cell of the grid:

```
<wj-flex-grid [itemsSource]="data">
  <template wjFlexGridCellTemplate [cellType]="'BottomLeft'">
    &#931;
  </template>
</wj-flex-grid>
```

NewCellTemplate

Defines a cell in a new row template. Must be a child of the **WjFlexGridColumn** component. Note that the **cell.item** property is undefined for this type of a cell. For example, this cell template shows a placeholder in the Date column's cell in the "new row" item:

```
<wj-flex-grid-column [header]="'Date'" [binding]="'date'">
  <template wjFlexGridCellTemplate [cellType]="'NewCellTemplate'">
    Enter a date here
  </template>
</wj-flex-grid-column>
```

Properties

- autoSizeRows
- cellOverflow
- forceFullEdit

Properties

- **autoSizeRows**

Gets or sets a value indicating whether the cell template will increase grid's default row height to accomodate cells content. Defaults to true.

Type
boolean

cellOverflow

Defines the **style.overflow** property value for cells.

Type
string

forceFullEdit

For cell edit templates, indicates whether cell editing forcibly starts in full edit mode, regardless of how the editing was initiated. In full edit mode pressing cursor keys don't finish editing. Defaults to true.

Type
boolean

WjFlexGridColumn Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid
Base Class
Column

Angular 2 component for the **Column** control.

The **wj-flex-grid-column** component must be contained in a **WjFlexGrid** component.

Use the **wj-flex-grid-column** component to add **Column** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexGridColumn** component is derived from the **Column** control and inherits all its properties, events and methods.

The **wj-flex-grid-column** component may contain a **WjFlexGridCellTemplate** child directive.

Constructor

- constructor

Properties

- aggregate
- align
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- asyncBindings
- binding
- collectionView
- cssClass
- currentSort
- dataMap
- dataType
- dropDownCssClass
- format
- grid
- header
- index
- initialized
- inputType
- isContentHtml
- isInitialized
- isReadOnly
- isRequired
- isSelected
- isVisible
- mask
- maxLength
- maxWidth
- minWidth
- name
- pos
- quickAutoSize
- renderSize
- renderWidth
- showDropDown
- size
- sortMemberPath
- visible
- visibleIndex
- width
- wjProperty
- wordWrap

Methods

- created
- getAlignment
- getIsRequired
- onPropertyChanged

Constructor

constructor

```
constructor(options?: any): Column
```

Initializes a new instance of the **Column** class.

Parameters

- **options**: **any** OPTIONAL
  Initialization options for the column.

Inherited From
Column
Returns
Column

Properties

aggregate

Gets or sets the **Aggregate** to display in the group header rows for the column.

Inherited From
Column
Type
Aggregate

align

Gets or sets the horizontal alignment of items in the column.

The default value for this property is null, which causes the grid to select the alignment automatically based on the column's **dataType** (numbers are right-aligned, Boolean values are centered, and other types are left-aligned).

If you want to override the default alignment, set this property to 'left', 'right', or 'center'.

Inherited From
Column
Type
string

allowDragging

Gets or sets a value that indicates whether the user can move the row or column to a new position with the mouse.

Inherited From
`RowCol`
Type
boolean

allowMerging

Gets or sets a value that indicates whether cells in the row or column can be merged.

Inherited From
`RowCol`
Type
boolean

allowResizing

Gets or sets a value that indicates whether the user can resize the row or column with the mouse.

Inherited From
`RowCol`
Type
boolean

allowSorting

Gets or sets a value that indicates whether the user can sort the column by clicking its header.

Inherited From
`Column`
Type
boolean

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

binding

Gets or sets the name of the property the column is bound to.

Inherited From
`Column`
Type
string

collectionView

Gets the **`ICollectionView`** bound to this row or column.

Inherited From
`RowCol`
Type
`ICollectionView`

cssClass

Gets or sets a CSS class name to use when rendering non-header cells in the row or column.

Inherited From
`RowCol`
Type
string

currentSort

Gets a string that describes the current sorting applied to the column. Possible values are '+' for ascending order, '-' for descending order, or null for unsorted columns.

Inherited From
`Column`
Type
string

dataMap

Gets or sets the **`DataMap`** used to convert raw values into display values for the column.

Columns with an associated **dataMap** show drop-down buttons that can be used for quick editing. If you do not want to show the drop-down buttons, set the column's **`showDropDown`** property to false.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`Column`
Type
`DataMap`

dataType

Gets or sets the type of value stored in the column.

Values are coerced into the proper type when editing the grid.

Inherited From
`Column`
Type
`DataType`

dropDownCssClass

Gets or sets a CSS class name to add to drop-downs in this column.

The drop-down buttons are shown only if the column has a **dataMap** set and is editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`Column`
Type
string

format

Gets or sets the format string used to convert raw values into display values for the column (see `Globalize`).

Inherited From
`Column`
Type
string

grid

Gets the `FlexGrid` that owns the row or column.

Inherited From
`RowCol`
Type
`FlexGrid`

header

Gets or sets the text displayed in the column header.

Inherited From
`Column`
Type
string

index

Gets the index of the row or column in the parent collection.

Inherited From
`RowCol`
Type
number

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

inputType

Gets or sets the "type" attribute of the HTML input element used to edit values in this column.

By default, this property is set to "tel" for numeric columns, and to "text" for all other non-boolean column types. The "tel" input type causes mobile devices to show a numeric keyboard that includes a negative sign and a decimal separator.

Use this property to change the default setting if the default does not work well for the current culture, device, or application. In these cases, try setting the property to "number" or simply "text."

Inherited From
`Column`
Type
string

isContentHtml

Gets or sets a value that indicates whether cells in this row or column contain HTML content rather than plain text.

Inherited From
`RowCol`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that indicates whether cells in the row or column can be edited.

Inherited From
`RowCol`
Type
boolean

isRequired

Gets or sets a value that determines whether values in the column are required.

By default, this property is set to null, which means values are required, but non-masked string columns may contain empty strings.

When set to true, values are required and empty strings are not allowed.

When set to false, null values and empty strings are allowed.

Inherited From
`Column`
Type
boolean

isSelected

Gets or sets a value that indicates whether the row or column is selected.

Inherited From
`RowCol`
Type
boolean

isVisible

Gets a value that indicates whether the row or column is visible and not collapsed.

This property is read-only. To change the visibility of a row or column, use the **`visible`** property instead.

Inherited From
`RowCol`
Type
boolean

mask

Gets or sets a mask to use while editing values in this column.

The mask format is the same used by the **`InputMask`** control.

If specified, the mask must be compatible with the value of the **`format`** property. For example, the mask '99/99/9999' can be used for entering dates formatted as 'MM/dd/yyyy'.

Inherited From
`Column`
Type
string

maxLength

Gets or sets the maximum number of characters that the can be entered into the cell.

Set this property to null to allow entry of any number of characters.

Inherited From
`Column`
Type
number

maxWidth

Gets or sets the maximum width of the column.

Inherited From
`Column`
Type
number

minWidth

Gets or sets the minimum width of the column.

Inherited From
`Column`
Type
number

name

Gets or sets the name of the column.

The column name can be used to retrieve the column using the **`getColumn`** method.

Inherited From
`Column`
Type
string

pos

Gets the position of the row or column.

Inherited From
`RowCol`
Type
number

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing this column.

Setting this property to false disables quick auto-sizing for this column. Setting it to true enables the feature, subject to the value of the grid's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for columns that display plain text and don't have templates.

Inherited From
`Column`
Type
boolean

renderSize

Gets the render size of the row or column. This property accounts for visibility, default size, and min and max sizes.

Inherited From
`RowCol`
Type
number

renderWidth

Gets the render width of the column.

The value returned takes into account the column's visibility, default size, and min and max sizes.

Inherited From
`Column`
Type
number

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in this column.

The drop-down buttons are shown only if the column has a **dataMap** set and is editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`Column`
Type
boolean

size

Gets or sets the size of the row or column. Setting this property to null or negative values causes the element to use the parent collection's default size.

Inherited From
`RowCol`
Type
number

sortMemberPath

Gets or sets the name of the property to use when sorting this column.

Use this property in cases where you want the sorting to be performed based on values other than the ones specified by the **`binding`** property.

Setting this property is null causes the grid to use the value of the **`binding`** property to sort the column.

Inherited From
`Column`
Type
string

visible

Gets or sets a value that indicates whether the row or column is visible.

Inherited From
`RowCol`
Type
boolean

visibleIndex

Gets the index of the row or column in the parent collection ignoring invisible elements (**`isVisible`**).

Inherited From
`RowCol`
Type
number

width

Gets or sets the width of the column.

Column widths may be positive numbers (sets the column width in pixels), null or negative numbers (uses the collection's default column width), or strings in the format '{number}*' (star sizing).

The star-sizing option performs a XAML-style dynamic sizing where column widths are proportional to the number before the star. For example, if a grid has three columns with widths "100", "*", and "3*", the first column will be 100 pixels wide, the second will take up 1/4th of the remaining space, and the last will take up the remaining 3/4ths of the remaining space.

Star-sizing allows you to define columns that automatically stretch to fill the width available. For example, set the width of the last column to "*" and it will automatically extend to fill the entire grid width so there's no empty space. You may also want to set the column's **`minWidth`** property to prevent the column from getting too narrow.

Inherited From
`Column`
Type
any

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'columns'.

Type
string

wordWrap

Gets or sets a value that indicates whether cells in the row or column wrap their content.

Inherited From
`RowCol`
Type
boolean

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

getAlignment

```
getAlignment(): string
```

Gets the actual column alignment.

Returns the value of the **align** property if it is not null, or selects the alignment based on the column's **dataType**.

Inherited From
Column
Returns
string

getIsRequired

```
getIsRequired(): boolean
```

Gets a value that determines whether the column is required.

Returns the value of the **isRequired** property if it is not null, or determines the required status based on the column's **dataType**.

By default, string columns are not required unless they have an associated **dataMap** or **mask**; all other data types are required.

Inherited From
Column
Returns
boolean

onPropertyChanged

```
onPropertyChanged(): void
```

Marks the owner list as dirty and refreshes the owner grid.

Inherited From
`RowCol`
Returns
void

CellTemplateType Enum

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid

Defines the type of cell on which a template is to be applied. This value is specified in the **cellType** attribute of the **WjFlexGridCellTemplate** directive.

Members

Name	Value	Description
Cell	0	Defines a regular (data) cell.
CellEdit	1	Defines a cell in edit mode.
ColumnHeader	2	Defines a column header cell.
RowHeader	3	Defines a row header cell.
RowHeaderEdit	4	Defines a row header cell in edit mode.
TopLeft	5	Defines a top left cell.
GroupHeader	6	Defines a group header cell in a group row.
Group	7	Defines a regular cell in a group row.
NewCellTemplate	8	Defines a cell in a new row template.
ColumnFooter	9	Defines a column footer cell.
BottomLeft	10	Defines a bottom left cell (at the intersection of the row header and column footer cells).

wijmo/wijmo.angular2.grid.filter Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.filter

Contains Angular 2 components for the **wijmo.grid.filter** module.

wijmo.angular2.grid.filter is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjFilter from 'wijmo/wijmo.angular2.grid.filter';
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjFilter.WjFlexGridFilter],
    template: `
      <wj-flex-grid [itemsSource]="data">
          <wj-flex-grid-filter [filterColumns]="['country', 'expenses']"></wj-flex-grid-filter>
      </wj-flex-grid>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFlexGridFilter

WjFlexGridFilter Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.filter
Base Class
FlexGridFilter

Angular 2 component for the **FlexGridFilter** control.

The **wj-flex-grid-filter** component must be contained in a **WjFlexGrid** component.

Use the **wj-flex-grid-filter** component to add **FlexGridFilter** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexGridFilter** component is derived from the **FlexGridFilter** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- defaultFilterType
- filterAppliedNg
- filterChangedNg
- filterChangingNg
- filterColumns
- filterDefinition
- grid
- initialized
- isInitialized
- showFilterIcons
- showSortButtons
- wjProperty

Methods

- apply
- clear
- closeEditor
- created
- editColumnFilter
- getColumnFilter
- onFilterApplied
- onFilterChanged
- onFilterChanging

Events

- filterApplied
- filterChanged
- filterChanging

Constructor

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridFilter
```

Initializes a new instance of the **FlexGridFilter** class.

Parameters

- **grid**: **FlexGrid**
  The **FlexGrid** to filter.

- **options**: **any** OPTIONAL
  Initialization options for the **FlexGridFilter**.

Inherited From
FlexGridFilter
Returns
FlexGridFilter

Properties

defaultFilterType

Gets or sets the default filter type to use.

This value can be overridden in filters for specific columns. For example, the code below creates a filter that filters by conditions on all columns except the "ByValue" column:

```
var f = new wijmo.grid.filter.FlexGridFilter(flex);
f.defaultFilterType = wijmo.grid.filter.FilterType.Condition;
var col = flex.columns.getColumn('ByValue'),
    cf = f.getColumnFilter(col);
cf.filterType = wijmo.grid.filter.FilterType.Value;
```

Inherited From
FlexGridFilter
Type
FilterType

filterAppliedNg

Angular (EventEmitter) version of the Wijmo **filterApplied** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **filterApplied** Wijmo event name.

Type
EventEmitter

filterChangedNg

Angular (EventEmitter) version of the Wijmo **filterChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **filterChanged** Wijmo event name.

Type
EventEmitter

filterChangingNg

Angular (EventEmitter) version of the Wijmo **filterChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **filterChanging** Wijmo event name.

Type
EventEmitter

filterColumns

Gets or sets an array containing the names or bindings of the columns that have filters.

Setting this property to null or to an empty array adds filters to all columns.

Inherited From
`FlexGridFilter`
Type
string[]

filterDefinition

Gets or sets the current filter definition as a JSON string.

Inherited From
`FlexGridFilter`
Type
string

grid

Gets a reference to the **`FlexGrid`** that owns this filter.

Inherited From
`FlexGridFilter`
Type
`FlexGrid`

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

showFilterIcons

Gets or sets a value indicating whether the **`FlexGridFilter`** adds filter editing buttons to the grid's column headers.

If you set this property to false, then you are responsible for providing a way for users to edit, clear, and apply the filters.

Inherited From
`FlexGridFilter`
Type
boolean

showSortButtons

Gets or sets a value indicating whether the filter editor should include sort buttons.

By default, the editor shows sort buttons like Excel does. But since users can sort columns by clicking their headers, sort buttons in the filter editor may not be desirable in some circumstances.

Inherited From
`FlexGridFilter`
Type
boolean

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

Methods

apply

```
apply(): void
```

Applies the current column filters to the grid.

Inherited From
`FlexGridFilter`
Returns
void

clear

```
clear(): void
```

Clears all column filters.

Inherited From
`FlexGridFilter`
Returns
void

closeEditor

```
closeEditor(): void
```

Closes the filter editor.

Inherited From
`FlexGridFilter`
Returns
void

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

editColumnFilter

```
editColumnFilter(col: any, ht?: HitTestInfo): void
```

Shows the filter editor for the given grid column.

Parameters

- **col**: **any**
  The **Column** that contains the filter to edit.

- **ht**: **HitTestInfo** OPTIONAL
  A **HitTestInfo** object containing the range of the cell that triggered the filter display.

Inherited From
FlexGridFilter
Returns
void

getColumnFilter

```
getColumnFilter(col: any, create?: boolean): ColumnFilter
```

Gets the filter for the given column.

Parameters

- **col**: **any**
  The **Column** that the filter applies to (or column name or index).

- **create**: **boolean** OPTIONAL
  Whether to create the filter if it does not exist.

Inherited From
FlexGridFilter
Returns
ColumnFilter

onFilterApplied

```
onFilterApplied(e?: EventArgs): void
```

Raises the **filterApplied** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGridFilter
Returns
void

onFilterChanged

```
onFilterChanged(e: CellRangeEventArgs): void
```

Raises the **filterChanged** event.

Parameters

- **e**: **CellRangeEventArgs**

Inherited From
FlexGridFilter
Returns
void

onFilterChanging

```
onFilterChanging(e: CellRangeEventArgs): void
```

Raises the **filterChanging** event.

Parameters

- **e**: **CellRangeEventArgs**

Inherited From
FlexGridFilter
Returns
void

Events

filterApplied

Occurs after the filter is applied.

Inherited From
`FlexGridFilter`
Arguments
`EventArgs`

filterChanged

Occurs after a column filter has been edited by the user.

Use the event parameters to determine the column that owns the filter and whether changes were applied or canceled.

Inherited From
`FlexGridFilter`
Arguments
`CellRangeEventArgs`

filterChanging

Occurs when a column filter is about to be edited by the user.

Use this event to customize the column filter if you want to override the default settings for the filter.

For example, the code below sets the operator used by the filter conditions to 'contains' if they are null:

```
filter.filterChanging.addHandler(function (s, e) {
  var cf = filter.getColumnFilter(e.col);
  if (!cf.valueFilter.isActive && cf.conditionFilter.condition1.operator == null) {
    cf.filterType = wijmo.grid.filter.FilterType.Condition;
    cf.conditionFilter.condition1.operator = wijmo.grid.filter.Operator.CT;
  }
});
```

Inherited From
`FlexGridFilter`
Arguments
`CellRangeEventArgs`

wijmo/wijmo.angular2.grid.grouppanel Module

File
wijmo.angular2.js
Module
wijmo/wijmo.angular2.grid.grouppanel

Contains Angular 2 components for the **wijmo.grid.grouppanel** module.

wijmo.angular2.grid.grouppanel is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjPanel from 'wijmo/wijmo.angular2.grid.grouppanel';
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjPanel.WjGroupPanel],
    template: `
      <wj-group-panel
          [grid]="flex"
          [placeholder]="'Drag columns here to create groups.'">
      </wj-group-panel>
      <wj-flex-grid #flex [itemsSource]="data">
      </wj-flex-grid>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjGroupPanel

WjGroupPanel Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.grouppanel
Base Class
GroupPanel

Angular 2 component for the **GroupPanel** control.

Use the **wj-group-panel** component to add **GroupPanel** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjGroupPanel** component is derived from the **GroupPanel** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- controlTemplate
- gotFocusNg
- grid
- hideGroupedColumns
- hostElement
- initialized
- isDisabled
- isInitialized
- isTouching
- isUpdating
- lostFocusNg
- maxGroups
- placeholder
- rightToLeft
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- refresh
- refreshAll
- removeEventListener

Events

- gotFocus
- lostFocus

Constructor

constructor

```
constructor(element: any, options?): GroupPanel
```

Initializes a new instance of the **GroupPanel** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  The JavaScript object containing initialization data for the control.

Inherited From
GroupPanel
Returns
GroupPanel

Properties

STATIC controlTemplate

Gets or sets the template used to instantiate **GroupPanel** controls.

Inherited From
GroupPanel
Type
any

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

grid

Gets or sets the **FlexGrid** that is connected to this **GroupPanel**.

Once a grid is connected to the panel, the panel displays the groups defined in the grid's data source. Users can drag grid columns into the panel to create new groups, drag groups within the panel to re-arrange the groups, or delete items in the panel to remove the groups.

Inherited From
GroupPanel
Type
FlexGrid

hideGroupedColumns

Gets or sets a value indicating whether the panel hides grouped columns in the owner grid.

The **FlexGrid** displays grouping information in row headers, so it is usually a good idea to hide grouped columns since they display redundant information.

Inherited From
GroupPanel
Type
boolean

hostElement

Gets the DOM element that is hosting the control.

Inherited From
Control
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

maxGroups

Gets or sets the maximum number of groups allowed.

Inherited From
`GroupPanel`
Type
number

placeholder

Gets or sets a string to display in the control when it contains no groups.

Inherited From
`GroupPanel`
Type
string

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

refresh

```
refresh(): void
```

Updates the panel to show the current groups.

Inherited From
GroupPanel
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

wijmo/wijmo.angular2.grid.detail Module

File
wijmo.angular2.js
Module
wijmo/wijmo.angular2.grid.detail

Contains Angular 2 components for the **wijmo.grid.detail** module.

wijmo.angular2.grid.detail is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjDetail from 'wijmo/wijmo.angular2.grid.detail';
import * as wjGrid from 'wijmo/wijmo.angular2.grid';

@Component({
    directives: [wjGrid.WjFlexGrid, wjDetail.WjFlexGridDetail],
    template: `
      <wj-flex-grid [itemsSource]="data">
          <template wjFlexGridDetail>
              Detail row content here...
          </template>
      </wj-flex-grid>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFlexGridDetail

WjFlexGridDetail Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.detail
Base Class
FlexGridDetailProvider

Angular 2 directive for **FlexGrid DetailRow** templates.

The **wj-flex-grid-detail** directive must be specified on a **<template>** template element contained in a **wj-flex-grid** component.

The **wj-flex-grid-detail** directive is derived from the **FlexGridDetailProvider** class that maintains detail rows visibility, with detail rows content defined as an arbitrary HTML fragment within the directive tag. The fragment may contain Angular 2 bindings, components and directives. The **row** and **item** template variables can be used in Angular 2 bindings that refer to the detail row's parent **Row** and **Row.dataItem** objects.

Constructor

- constructor

Properties

- createDetailCell
- detailVisibilityMode
- disposeDetailCell
- grid
- initialized
- isAnimated
- isInitialized
- maxHeight
- rowHasDetail

Methods

- created
- getDetailRow
- hideDetail
- isDetailAvailable
- isDetailVisible
- showDetail

Constructor

constructor

```
constructor(grid: FlexGrid, options?: any): FlexGridDetailProvider
```

Initializes a new instance of the **FlexGridDetailProvider** class.

Parameters

- **grid**: **FlexGrid**
  FlexGrid that will receive detail rows.

- **options**: **any** OPTIONAL
  Initialization options for the new **FlexGridDetailProvider**.

Inherited From
FlexGridDetailProvider
Returns
FlexGridDetailProvider

Properties

createDetailCell

Gets or sets the callback function that creates detail cells.

The callback function takes a **Row** as a parameter and returns an HTML element representing the row details. For example:

```
// create detail cells for a given row
dp.createDetailCell = function (row) {
  var cell = document.createElement('div');
  var detailGrid = new wijmo.grid.FlexGrid(cell, {
    itemsSource: getProducts(row.dataItem.CategoryID),
    headersVisibility: wijmo.grid.HeadersVisibility.Column
  });
  return cell;
};
```

Inherited From
FlexGridDetailProvider
Type
Function

detailVisibilityMode

Gets or sets a value that determines when row details are displayed.

Inherited From
`FlexGridDetailProvider`
Type
`DetailVisibilityMode`

disposeDetailCell

Gets or sets the callback function that disposes of detail cells.

The callback function takes a **`Row`** as a parameter and disposes of any resources associated with the detail cell.

This function is optional. Use it in cases where the **`createDetailCell`** function allocates resources that are not automatically garbage-collected.

Inherited From
`FlexGridDetailProvider`
Type
Function

grid

Gets the **`FlexGrid`** that owns this **`FlexGridDetailProvider`**.

Inherited From
`FlexGridDetailProvider`
Type
`FlexGrid`

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isAnimated

Gets or sets a value that indicates whether to use animation when showing row details.

Inherited From
`FlexGridDetailProvider`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

maxHeight

Gets or sets the maximum height of the detail rows, in pixels.

Inherited From
`FlexGridDetailProvider`
Type
number

rowHasDetail

Gets or sets the callback function that determines whether a row has details.

The callback function takes a **Row** as a parameter and returns a boolean value that indicates whether the row has details. For example:

```
// remove details from items with odd CategoryID
dp.rowHasDetail = function (row) {
  return row.dataItem.CategoryID % 2 == 0;
};
```

Setting this property to null indicates all rows have details.

Inherited From
`FlexGridDetailProvider`
Type
Function

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

getDetailRow

```
getDetailRow(row: any): DetailRow
```

Gets the detail row associated with a given grid row.

Parameters

- **row**: **any**
  Row or index of the row to investigate.

Inherited From
`FlexGridDetailProvider`
Returns
`DetailRow`

hideDetail

```
hideDetail(row?: any): void
```

Hides the detail row for a given row.

Parameters

- **row**: **any** OPTIONAL
  Row or index of the row that will have its details hidden. This parameter is optional. If not provided, all detail rows are hidden.

Inherited From
`FlexGridDetailProvider`
Returns
void

isDetailAvailable

```
isDetailAvailable(row: any): boolean
```

Gets a value that determines if a row has details to show.

Parameters

- **row**: **any**
  Row or index of the row to investigate.

Inherited From
`FlexGridDetailProvider`
Returns
boolean

isDetailVisible

```
isDetailVisible(row: any): boolean
```

Gets a value that determines if a row's details are visible.

Parameters

- **row**: **any**
  Row or index of the row to investigate.

Inherited From
`FlexGridDetailProvider`
Returns
boolean

showDetail

```
showDetail(row: any, hideOthers?: boolean): void
```

Shows the detail row for a given row.

Parameters

- **row**: **any**
  Row or index of the row that will have its details shown.

- **hideOthers**: **boolean** OPTIONAL
  Whether to hide details for all other rows.

Inherited From
`FlexGridDetailProvider`
Returns
void

wijmo/wijmo.angular2.grid.multirow Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.multirow

Contains Angular 2 components for the **wijmo.grid.multirow** module.

wijmo.angular2.grid.multirow is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjMultiRow from 'wijmo/wijmo.angular2.grid.multirow';

@Component({
    directives: [wjMultiRow.WjMultiRow],
    template: `<wj-multi-row></wj-multi-row>`,
    selector: 'my-cmp',
})
export class MyCmp {
}
```

Classes

- WjMultiRow

WjMultiRow Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.multirow
Base Class
MultiRow

Angular 2 component for the **MultiRow** control.

Use the **wj-multi-row** component to add **MultiRow** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjMultiRow** component is derived from the **MultiRow** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- autoClipboard
- autoGenerateColumns
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells
- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- centerHeadersVertically
- childItemsPath
- clientSize
- cloneFrozenCells
- collapsedHeaders
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- copiedNg
- copyingNg
- deferResizing
- deletedRowNg
- deletingRowNg
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowNg
- draggingRowOverNg
- editableCollectionView
- editRange
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isInitialized
- isReadOnly
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab
- layoutDefinition
- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager
- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft
- rowAddedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg

- rowEditStartingNg
- rowHeaderPath
- rowHeaders
- rows
- rowsPerItem
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- showAlternatingRows
- showDropDown
- showErrors
- showGroups
- showHeaderCollapseButton
- showMarquee
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- collapseGroupsToLevel
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- finishEditing
- focus
- getBindingColumn
- getCellBoundingRect
- getCellData
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- isRangeValid
- onAutoSizedColumn
- onAutoSizedRow
- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowOver
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onResizedColumn
- onResizedRow

- onResizingColumn
- onResizingRow
- onRowAdded
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectionChanged
- onSelectionChanging
- onSortedColumn
- onSortingColumn
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- refresh
- refreshAll
- refreshCells
- removeEventListener
- scrollIntoView
- select
- setCellData
- setClipString
- startEditing
- toggleDropDownList

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowOver
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging
- itemsSourceChanged
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectionChanged
- selectionChanging
- sortedColumn
- sortingColumn
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): MultiRow
```

Initializes a new instance of the **MultiRow** class.

In most cases, the **options** parameter will include the value for the **layoutDefinition** property.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Inherited From
MultiRow
Returns
MultiRow

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Inherited From
FlexGrid
Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **isReadOnly** property is set to true.

Inherited From
FlexGrid
Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Inherited From
`FlexGrid`
Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Inherited From
`FlexGrid`
Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Inherited From
`FlexGrid`
Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Inherited From
`FlexGrid`
Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Inherited From
`FlexGrid`
Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **itemsSource**.

The column generation depends on the **itemsSource** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **autoGenerateColumns** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Inherited From
FlexGrid
Type
boolean

autoSizedColumnNg

Angular (EventEmitter) version of the Wijmo **autoSizedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizedColumn** Wijmo event name.

Type
EventEmitter

autoSizedRowNg

Angular (EventEmitter) version of the Wijmo **autoSizedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizedRow** Wijmo event name.

Type
EventEmitter

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Inherited From
`FlexGrid`
Type
`AutoSizeMode`

autoSizingColumnNg

Angular (EventEmitter) version of the Wijmo **autoSizingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizingColumn** Wijmo event name.

Type
EventEmitter

autoSizingRowNg

Angular (EventEmitter) version of the Wijmo **autoSizingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizingRow** Wijmo event name.

Type
EventEmitter

beginningEditNg

Angular (EventEmitter) version of the Wijmo **beginningEdit** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **beginningEdit** Wijmo event name.

Type
EventEmitter

bottomLeftCells

Gets the **GridPanel** that contains the bottom left cells.

The **bottomLeftCells** panel appears below the row headers, to the left of the **columnFooters** panel.

Inherited From
FlexGrid
Type
GridPanel

cellEditEndedNg

Angular (EventEmitter) version of the Wijmo **cellEditEnded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **cellEditEnded** Wijmo event name.

Type
EventEmitter

cellEditEndingNg

Angular (EventEmitter) version of the Wijmo **cellEditEnding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **cellEditEnding** Wijmo event name.

Type
EventEmitter

cellFactory

Gets or sets the **CellFactory** that creates and updates cells for this grid.

Inherited From
FlexGrid
Type
CellFactory

cells

Gets the **`GridPanel`** that contains the data cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

centerHeadersVertically

Gets or sets a value that determines whether the content of cells that span multiple rows should be vertically centered.

Inherited From
`MultiRow`
Type
boolean

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Inherited From
`FlexGrid`
Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Inherited From
`FlexGrid`
Type
`Size`

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Inherited From
`FlexGrid`
Type
boolean

collapsedHeaders

Gets or sets a value that determines whether column headers should be collapsed and displayed as a single row containing the group headers.

If you set the `collapsedHeaders` property to **true**, remember to set the **header** property of every group in order to avoid empty header cells.

Setting the `collapsedHeaders` property to **null** causes the grid to show all header information (groups and columns). In this case, the first row will show the group headers and the remaining rows will show the individual column headers.

Inherited From
`MultiRow`
Type
boolean

collectionView

Gets the `ICollectionView` that contains the grid data.

Inherited From
`FlexGrid`
Type
`ICollectionView`

columnFooters

Gets the **GridPanel** that contains the column footer cells.

The **columnFooters** panel appears below the grid cells, to the right of the **bottomLeftCells** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **columnFooters** panel to display summary data for columns that have the **aggregate** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Inherited From
FlexGrid
Type
GridPanel

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Inherited From
FlexGrid
Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Inherited From
`FlexGrid`
Type
string

columns

Gets the grid's column collection.

Inherited From
`FlexGrid`
Type
`ColumnCollection`

controlRect

Gets the bounding rectangle of the control in page coordinates.

Inherited From
`FlexGrid`
Type
`Rect`

STATIC controlTemplate

Gets or sets the template used to instantiate `FlexGrid` controls.

Inherited From
`FlexGrid`
Type
any

copiedNg

Angular (EventEmitter) version of the Wijmo **copied** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **copied** Wijmo event name.

Type
EventEmitter

copyingNg

Angular (EventEmitter) version of the Wijmo **copying** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **copying** Wijmo event name.

Type
EventEmitter

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **`deferResizing`** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Inherited From
`FlexGrid`
Type
boolean

deletedRowNg

Angular (EventEmitter) version of the Wijmo **deletedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **deletedRow** Wijmo event name.

Type
EventEmitter

deletingRowNg

Angular (EventEmitter) version of the Wijmo **deletingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **deletingRow** Wijmo event name.

Type
EventEmitter

draggedColumnNg

Angular (EventEmitter) version of the Wijmo **draggedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggedColumn** Wijmo event name.

Type
EventEmitter

draggedRowNg

Angular (EventEmitter) version of the Wijmo **draggedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggedRow** Wijmo event name.

Type
EventEmitter

draggingColumnNg

Angular (EventEmitter) version of the Wijmo **draggingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingColumn** Wijmo event name.

Type
EventEmitter

draggingColumnOverNg

Angular (EventEmitter) version of the Wijmo **draggingColumnOver** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingColumnOver** Wijmo event name.

Type
EventEmitter

draggingRowNg

Angular (EventEmitter) version of the Wijmo **draggingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRow** Wijmo event name.

Type
EventEmitter

draggingRowOverNg

Angular (EventEmitter) version of the Wijmo **draggingRowOver** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRowOver** Wijmo event name.

Type
EventEmitter

editableCollectionView

Gets the **`IEditableCollectionView`** that contains the grid data.

Inherited From
`FlexGrid`
Type
`IEditableCollectionView`

editRange

Gets a **`CellRange`** that identifies the cell currently being edited.

Inherited From
`FlexGrid`
Type
`CellRange`

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

groupCollapsedChangedNg

Angular (EventEmitter) version of the Wijmo **groupCollapsedChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **groupCollapsedChanged** Wijmo event name.

Type
EventEmitter

groupCollapsedChangingNg

Angular (EventEmitter) version of the Wijmo **groupCollapsedChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **groupCollapsedChanging** Wijmo event name.

Type
EventEmitter

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Inherited From
FlexGrid
Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Inherited From
FlexGrid
Type
HeadersVisibility

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Inherited From
`FlexGrid`
Type
boolean

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Inherited From
`FlexGrid`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Inherited From
FlexGrid
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** that contains items shown on the grid.

Inherited From
`FlexGrid`
Type
any

itemsSourceChangedNg

Angular (EventEmitter) version of the Wijmo **itemsSourceChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **itemsSourceChanged** Wijmo event name.

Type
EventEmitter

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **`getError`** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Inherited From
`FlexGrid`
Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **`MoveDown`**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Inherited From
`FlexGrid`
Type
`KeyAction`

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **`None`**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **`Cycle`**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **`CycleOut`** setting that causes the selection to move through the cells (as **`Cycle`**), and then on to the next/previous control on the page when the last or first cells are selected.

Inherited From
`FlexGrid`
Type
`KeyAction`

layoutDefinition

Gets or sets an array that defines the layout of the rows used to display each data item.

The array contains a list of cell group objects which have the following properties:

- **header**: Group header (shown when the headers are collapsed)
- **colspan**: Number of grid columns spanned by the group
- **cells**: Array of cell objects, which extend **Column** with a **colspan** property.

When the **`layoutDefinition`** property is set, the grid scans the cells in each group as follows:

1. The grid calculates the colspan of the group either as group's own colspan or as span of the widest cell in the group, whichever is wider.
2. If the cell fits the current row within the group, it is added to the current row.
3. If it doesn't fit, it is added to a new row.

When all groups are ready, the grid calculates the number of rows per record to the maximum rowspan of all groups, and adds rows to each group to pad their height as needed.

This scheme is simple and flexible. For example:

```
{ header: 'Group 1', cells: [{ binding: 'c1' }, { bnding: 'c2'}, { binding: 'c3' }]}
```

The group has colspan 1, so there will be one cell per column. The result is:

```
| C1 |
| C2 |
| C3 |
```

To create a group with two columns, set **colspan** property of the group:

```
{ header: 'Group 1', colspan: 2, cells:[{ binding: 'c1' }, { binding: 'c2'}, { binding: 'c3' }]}
```

The cells will wrap as follows:

```
| C1  | C2 |
| C3       |
```

Note that the last cell spans two columns (to fill the group).

You can also specify the colspan on individual cells rather than on the group:

```
{ header: 'Group 1', cells: [{binding: 'c1', colspan: 2 }, { bnding: 'c2'}, { binding: 'c3' }]}
```

Now the first cell has colspan 2, so the result is:

```
| C1       |
| C2 |  C3 |
```

Because cells extend the **`Column`** class, you can add all the usual **`Column`** properties to any cells:

```
{ header: 'Group 1', cells: [
   { binding: 'c1', colspan: 2 },
   { bnding: 'c2'},
   { binding: 'c3', format: 'n0', required: false, etc... }
]}
```

Inherited From
`MultiRow`
Type
any[]

loadedRowsNg

Angular (EventEmitter) version of the Wijmo **loadedRows** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loadedRows** Wijmo event name.

Type
EventEmitter

loadingRowsNg

Angular (EventEmitter) version of the Wijmo **loadingRows** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loadingRows** Wijmo event name.

Type
EventEmitter

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mergeManager

Gets or sets the **`MergeManager`** object responsible for determining how cells should be merged.

Inherited From
`FlexGrid`
Type
`MergeManager`

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **`newRowAtTop`** property to true, and you want the new row template to remain visible at all times, set the **`frozenRows`** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **`allowAddNew`** property is set to true and if the **`itemsSource`** object supports adding new items.

Inherited From
`FlexGrid`
Type
boolean

pastedCellNg

Angular (EventEmitter) version of the Wijmo **pastedCell** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pastedCell** Wijmo event name.

Type
EventEmitter

pastedNg

Angular (EventEmitter) version of the Wijmo **pasted** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pasted** Wijmo event name.

Type
EventEmitter

pastingCellNg

Angular (EventEmitter) version of the Wijmo **pastingCell** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pastingCell** Wijmo event name.

Type
EventEmitter

pastingNg

Angular (EventEmitter) version of the Wijmo **pasting** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pasting** Wijmo event name.

Type
EventEmitter

prepareCellForEditNg

Angular (EventEmitter) version of the Wijmo **prepareCellForEdit** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **prepareCellForEdit** Wijmo event name.

Type
EventEmitter

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **`preserveOutlineState`** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Inherited From
`FlexGrid`
Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Inherited From
`FlexGrid`
Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Inherited From
`FlexGrid`
Type
boolean

resizedColumnNg

Angular (EventEmitter) version of the Wijmo **resizedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizedColumn** Wijmo event name.

Type
EventEmitter

resizedRowNg

Angular (EventEmitter) version of the Wijmo **resizedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizedRow** Wijmo event name.

Type
EventEmitter

resizingColumnNg

Angular (EventEmitter) version of the Wijmo **resizingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizingColumn** Wijmo event name.

Type
EventEmitter

resizingRowNg

Angular (EventEmitter) version of the Wijmo **resizingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizingRow** Wijmo event name.

Type
EventEmitter

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowAddedNg

Angular (EventEmitter) version of the Wijmo **rowAdded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowAdded** Wijmo event name.

Type
EventEmitter

rowEditEndedNg

Angular (EventEmitter) version of the Wijmo **rowEditEnded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditEnded** Wijmo event name.

Type
EventEmitter

rowEditEndingNg

Angular (EventEmitter) version of the Wijmo **rowEditEnding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditEnding** Wijmo event name.

Type
EventEmitter

rowEditStartedNg

Angular (EventEmitter) version of the Wijmo **rowEditStarted** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditStarted** Wijmo event name.

Type
EventEmitter

rowEditStartingNg

Angular (EventEmitter) version of the Wijmo **rowEditStarting** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditStarting** Wijmo event name.

Type
EventEmitter

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Inherited From
`FlexGrid`
Type
string

rowHeaders

Gets the **GridPanel** that contains the row header cells.

Inherited From
FlexGrid
Type
GridPanel

rows

Gets the grid's row collection.

Inherited From
FlexGrid
Type
RowCollection

rowsPerItem

Gets the number of rows used to display each item.

This value is calculated automatically based on the value of the **layoutDefinition** property.

Inherited From
MultiRow
Type
number

scrollPosition

Gets or sets a **Point** that represents the value of the grid's scrollbars.

Inherited From
FlexGrid
Type
Point

scrollPositionChangedNg

Angular (EventEmitter) version of the Wijmo **scrollPositionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **scrollPositionChanged** Wijmo event name.

Type
EventEmitter

scrollSize

Gets the size of the grid content in pixels.

Inherited From
`FlexGrid`
Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selection

Gets or sets the current selection.

Inherited From
`FlexGrid`
Type
`CellRange`

selectionChangedNg

Angular (EventEmitter) version of the Wijmo **selectionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanged** Wijmo event name.

Type
EventEmitter

selectionChangingNg

Angular (EventEmitter) version of the Wijmo **selectionChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanging** Wijmo event name.

Type
EventEmitter

selectionMode

Gets or sets the current selection mode.

Inherited From
`FlexGrid`
Type
`SelectionMode`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Inherited From
`FlexGrid`
Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **`showDropDown`** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`FlexGrid`
Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **`itemValidator`** property or the **`getError`** property on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showHeaderCollapseButton

Gets or sets a value that determines whether the grid should display a button in the column header panel to allow users to collapse and expand the column headers.

If the button is visible, clicking on it will cause the grid to toggle the value of the **collapsedHeaders** property.

Inherited From
`MultiRow`
Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Inherited From
`FlexGrid`
Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

sortedColumnNg

Angular (EventEmitter) version of the Wijmo **sortedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sortedColumn** Wijmo event name.

Type
EventEmitter

sortingColumnNg

Angular (EventEmitter) version of the Wijmo **sortingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sortingColumn** Wijmo event name.

Type
EventEmitter

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Inherited From
`FlexGrid`
Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Inherited From
`FlexGrid`
Type
boolean

topLeftCells

Gets the `GridPanel` that contains the top left cells (to the left of the column headers).

Inherited From
`FlexGrid`
Type
`GridPanel`

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Inherited From
`FlexGrid`
Type
number

updatedLayoutNg

Angular (EventEmitter) version of the Wijmo **updatedLayout** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatedLayout** Wijmo event name.

Type
EventEmitter

updatedViewNg

Angular (EventEmitter) version of the Wijmo **updatedView** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatedView** Wijmo event name.

Type
EventEmitter

updatingLayoutNg

Angular (EventEmitter) version of the Wijmo **updatingLayout** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatingLayout** Wijmo event name.

Type
EventEmitter

updatingViewNg

Angular (EventEmitter) version of the Wijmo **updatingView** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatingView** Wijmo event name.

Type
EventEmitter

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **`getError`** method on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

viewRange

Gets the range of cells currently in view.

Inherited From
`FlexGrid`
Type
`CellRange`

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Inherited From
`FlexGrid`
Type
number

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.

- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`Control`
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
FlexGrid
Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Inherited From
FlexGrid
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
`Control`
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Inherited From
`FlexGrid`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
`Control`
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **`beginUpdate`**.

Inherited From
`Control`
Returns
void

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Inherited From
FlexGrid
Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Inherited From
FlexGrid
Returns
void

getBindingColumn

```
getBindingColumn(p: GridPanel, r: number, c: number): Column
```

Gets the **Column** object used to bind a data item to a grid cell.

Parameters

- **p**: **GridPanel**
  GridPanel that contains the cell.

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
MultiRow
Returns
Column

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
`FlexGrid`
Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Inherited From
FlexGrid
Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Inherited From
`MultiRow`
Returns
`Column`

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getMergedRange

```
getMergedRange(p: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**.

Parameters

- **p**: **GridPanel**
  The **GridPanel** that contains the range.
- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Inherited From
FlexGrid
Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.
- **c**: **number**
  Column index of the cell to inspect.

Inherited From
FlexGrid
Returns
SelectedState

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Inherited From
FlexGrid
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Inherited From
FlexGrid
Returns
boolean

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  `CellRangeEventArgs` that contains the event data.

Inherited From
`FlexGrid`
Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the grid layout and content, or just the content.

Inherited From
FlexGrid
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.

- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.

- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Inherited From
FlexGrid
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.
- **c**: **number**
  Index of the column to scroll into view.

Inherited From
FlexGrid
Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

Parameters

- **rng**: **any**
  Range to select.
- **show**: **any** OPTIONAL
  Whether to scroll the new selection into view.

Inherited From
FlexGrid
Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean, invalidate?: boolean): boolean
```

Sets the value of a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.
- **invalidate**: **boolean** OPTIONAL
  Whether to invalidate the grid to show the change.

Inherited From
`FlexGrid`
Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Hidden rows and columns are skipped.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexGrid
Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Inherited From
FlexGrid
Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Inherited From
`FlexGrid`
Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Inherited From
`FlexGrid`
Arguments
`CellEditEndingEventArgs`

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **allowDelete** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **allowDelete** property).

The event handler may cancel the row deletion.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedColumn

Occurs when the user finishes dragging a column.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

draggedRow

Occurs when the user finishes dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRowOver

Occurs as the user drags a row to a new position.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **`itemFormatter`** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Inherited From
`FlexGrid`
Arguments
`FormatItemEventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **`autoClipboard`** property).

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

resizedColumn

Occurs when the user finishes resizing a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizedRow

Occurs when the user finishes resizing rows.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **`rowEditStarted`** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarted

Occurs after a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

selectionChanged

Occurs after selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

updatedLayout

Occurs after the grid has updated its internal layout.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

updatingLayout

Occurs before the grid updates its internal layout.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

wijmo/wijmo.angular2.grid.sheet Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.sheet

Contains Angular 2 components for the **wijmo.grid.sheet** module.

wijmo.angular2.grid.sheet is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjSheet from 'wijmo/wijmo.angular2.grid.sheet';

@Component({
    directives: [wjSheet.WjFlexSheet],
    template: `<wj-flex-sheet></wj-flex-sheet>`,
    selector: 'my-cmp',
})
export class MyCmp {
}
```

Classes

- WjFlexSheet
- WjSheet

WjFlexSheet Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.sheet
Base Class
FlexSheet

Angular 2 component for the **FlexSheet** control.

Use the **wj-flex-sheet** component to add **FlexSheet** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexSheet** component is derived from the **FlexSheet** control and inherits all its properties, events and methods.

The **wj-flex-sheet** component may contain a **WjSheet** child component.

Constructor

- constructor

Properties

- activeEditor
- allowAddNew
- allowDelete
- allowDragging
- allowMerging
- allowResizing
- allowSorting
- asyncBindings
- autoClipboard
- autoGenerateColumns
- autoSizedColumnNg
- autoSizedRowNg
- autoSizeMode
- autoSizingColumnNg
- autoSizingRowNg
- beginningEditNg
- bottomLeftCells
- cellEditEndedNg
- cellEditEndingNg
- cellFactory
- cells
- childItemsPath
- clientSize
- cloneFrozenCells
- collectionView
- columnFooters
- columnHeaders
- columnLayout
- columns
- controlRect
- controlTemplate
- copiedNg
- copyingNg
- deferResizing
- definedNames
- deletedRowNg
- deletingRowNg
- draggedColumnNg
- draggedRowNg
- draggingColumnNg
- draggingColumnOverNg
- draggingRowColumnNg
- draggingRowNg
- draggingRowOverNg
- droppingRowColumnNg
- editableCollectionView
- editRange
- formatItemNg
- frozenColumns
- frozenRows
- gotFocusNg
- groupCollapsedChangedNg
- groupCollapsedChangingNg
- groupHeaderFormat
- headersVisibility
- hostElement
- imeEnabled
- initialized
- isDisabled
- isFunctionListOpen
- isInitialized
- isReadOnly
- isTabHolderVisible
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- itemsSourceChangedNg
- itemValidator
- keyActionEnter
- keyActionTab
- loadedNg
- loadedRowsNg
- loadingRowsNg
- lostFocusNg
- mergeManager
- newRowAtTop
- pastedCellNg
- pastedNg
- pastingCellNg
- pastingNg
- prepareCellForEditNg
- preserveOutlineState
- preserveSelectedState
- quickAutoSize
- resizedColumnNg
- resizedRowNg
- resizingColumnNg
- resizingRowNg
- rightToLeft

- rowAddedNg
- rowEditEndedNg
- rowEditEndingNg
- rowEditStartedNg
- rowEditStartingNg
- rowHeaderPath
- rowHeaders
- rows
- scrollPosition
- scrollPositionChangedNg
- scrollSize
- selectedItems
- selectedRows
- selectedSheet
- selectedSheetChangedNg
- selectedSheetIndex
- selection
- selectionChangedNg
- selectionChangingNg
- selectionMode
- sheetClearedNg
- sheets
- showAlternatingRows
- showDropDown
- showErrors
- showFilterIcons
- showGroups
- showMarquee
- showSelectedHeaders
- showSort
- sortedColumnNg
- sortingColumnNg
- sortManager
- sortRowIndex
- stickyHeaders
- topLeftCells
- treeIndent
- undoStack
- unknownFunctionNg
- updatedLayoutNg
- updatedViewNg
- updatingLayoutNg
- updatingViewNg
- validateEdits
- viewRange
- virtualizationThreshold
- wjModelProperty

Methods

- addBoundSheet
- addEventListener
- addFunction
- addUnboundSheet
- applyCellsStyle
- applyFunctionToCell
- applyTemplate
- autoSizeColumn
- autoSizeColumns
- autoSizeRow
- autoSizeRows
- beginUpdate
- canEditCell
- clear
- collapseGroupsToLevel
- containsFocus
- convertNumberToAlpha
- created
- deferUpdate
- deleteColumns
- deleteRows
- dispose
- disposeAll
- endUpdate
- evaluate
- finishEditing
- focus
- freezeAtCursor
- getCellBoundingRect
- getCellData
- getCellValue
- getClipString
- getColumn
- getControl
- getMergedRange
- getSelectedState
- getSelectionFormatState
- getTemplate
- hideFunctionList
- hitTest
- initialize
- insertColumns
- insertRows
- invalidate
- invalidateAll
- isRangeValid
- load
- loadAsync
- mergeRange
- onAutoSizedColumn
- onAutoSizedRow

- onAutoSizingColumn
- onAutoSizingRow
- onBeginningEdit
- onCellEditEnded
- onCellEditEnding
- onColumnChanged
- onCopied
- onCopying
- onDeletedRow
- onDeletingRow
- onDraggedColumn
- onDraggedRow
- onDraggingColumn
- onDraggingColumnOver
- onDraggingRow
- onDraggingRowColumn
- onDraggingRowOver
- onDroppingRowColumn
- onFormatItem
- onGotFocus
- onGroupCollapsedChanged
- onGroupCollapsedChanging
- onItemsSourceChanged
- onLoaded
- onLoadedRows
- onLoadingRows
- onLostFocus
- onPasted
- onPastedCell
- onPasting
- onPastingCell
- onPrepareCellForEdit
- onPrepareChangingColumn
- onPrepareChangingRow
- onResizedColumn
- onResizedRow
- onResizingColumn
- onResizingRow
- onRowAdded
- onRowChanged
- onRowEditEnded
- onRowEditEnding
- onRowEditStarted
- onRowEditStarting
- onScrollPositionChanged
- onSelectedSheetChanged
- onSelectionChanged
- onSelectionChanging
- onSheetCleared
- onSortedColumn
- onSortingColumn
- onUnknownFunction
- onUpdatedLayout
- onUpdatedView
- onUpdatingLayout
- onUpdatingView
- redo
- refresh
- refreshAll
- refreshCells
- removeEventListener
- save
- saveAsync
- scrollIntoView
- select
- selectNextFunction
- selectPreviousFunction
- setCellData
- setClipString
- showColumnFilter
- showFunctionList
- startEditing
- toggleDropDownList
- undo

Events

- autoSizedColumn
- autoSizedRow
- autoSizingColumn
- autoSizingRow
- beginningEdit
- cellEditEnded
- cellEditEnding
- columnChanged
- copied
- copying
- deletedRow
- deletingRow
- draggedColumn
- draggedRow
- draggingColumn
- draggingColumnOver
- draggingRow
- draggingRowColumn
- draggingRowOver
- droppingRowColumn
- formatItem
- gotFocus
- groupCollapsedChanged
- groupCollapsedChanging

- itemsSourceChanged
- loaded
- loadedRows
- loadingRows
- lostFocus
- pasted
- pastedCell
- pasting
- pastingCell
- prepareCellForEdit
- prepareChangingColumn
- prepareChangingRow
- resizedColumn
- resizedRow
- resizingColumn
- resizingRow
- rowAdded
- rowChanged
- rowEditEnded
- rowEditEnding
- rowEditStarted
- rowEditStarting
- scrollPositionChanged
- selectedSheetChanged
- selectionChanged
- selectionChanging
- sheetCleared
- sortedColumn
- sortingColumn
- unknownFunction
- updatedLayout
- updatedView
- updatingLayout
- updatingView

Constructor

constructor

```
constructor(element: any, options?): FlexSheet
```

Initializes a new instance of the **FlexSheet** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a jQuery selector (e.g. '#theCtrl').

- **options**: OPTIONAL
  JavaScript object containing initialization data for the control.

Inherited From
FlexSheet
Returns
FlexSheet

Properties

activeEditor

Gets the **HTMLInputElement** that represents the cell editor currently active.

Inherited From
`FlexGrid`
Type
HTMLInputElement

allowAddNew

Gets or sets a value that indicates whether the grid should provide a new row template so users can add items to the source collection.

The new row template will not be displayed if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDelete

Gets or sets a value that indicates whether the grid should delete selected rows when the user presses the Delete key.

Selected rows will not be deleted if the **`isReadOnly`** property is set to true.

Inherited From
`FlexGrid`
Type
boolean

allowDragging

Gets or sets a value that determines whether users are allowed to drag rows and/or columns with the mouse.

Inherited From
`FlexGrid`
Type
`AllowDragging`

allowMerging

Gets or sets which parts of the grid provide cell merging.

Inherited From
`FlexGrid`
Type
`AllowMerging`

allowResizing

Gets or sets a value that determines whether users may resize rows and/or columns with the mouse.

If resizing is enabled, users can resize columns by dragging the right edge of column header cells, or rows by dragging the bottom edge of row header cells.

Users may also double-click the edge of the header cells to automatically resize rows and columns to fit their content. The auto-size behavior can be customized using the **`autoSizeMode`** property.

Inherited From
`FlexGrid`
Type
`AllowResizing`

allowSorting

Gets or sets a value that determines whether users are allowed to sort columns by clicking the column header cells.

Inherited From
`FlexGrid`
Type
boolean

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.`asyncBindings`** property description for details.

Type
boolean

autoClipboard

Gets or sets a value that determines whether the grid should handle clipboard shortcuts.

The clipboard shortcuts are as follows:

ctrl+C, ctrl+Ins
Copy grid selection to clipboard.
ctrl+V, shift+Ins
Paste clipboard text to grid selection.

Only visible rows and columns are included in clipboard operations.

Read-only cells are not affected by paste operations.

Inherited From
`FlexGrid`
Type
boolean

autoGenerateColumns

Gets or sets a value that determines whether the grid should generate columns automatically based on the **`itemsSource`**.

The column generation depends on the **`itemsSource`** property containing at least one item. This data item is inspected and a column is created and bound to each property that contains a primitive value (number, string, Boolean, or Date).

Properties set to null do not generate columns, because the grid would have no way of guessing the appropriate type. In this type of scenario, you should set the **`autoGenerateColumns`** property to false and create the columns explicitly. For example:

```
var grid = new wijmo.grid.FlexGrid('#theGrid', {
  autoGenerateColumns: false, // data items may contain null values
  columns: [                  // so define columns explicitly
    { binding: 'name', header: 'Name', type: 'String' },
    { binding: 'amount', header: 'Amount', type: 'Number' },
    { binding: 'date', header: 'Date', type: 'Date' },
    { binding: 'active', header: 'Active', type: 'Boolean' }
  ],
  itemsSource: customers
});
```

Inherited From
`FlexGrid`
Type
boolean

autoSizedColumnNg

Angular (EventEmitter) version of the Wijmo **autoSizedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizedColumn** Wijmo event name.

Type
EventEmitter

autoSizedRowNg

Angular (EventEmitter) version of the Wijmo **autoSizedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizedRow** Wijmo event name.

Type
EventEmitter

autoSizeMode

Gets or sets which cells should be taken into account when auto-sizing a row or column.

This property controls what happens when users double-click the edge of a column header.

By default, the grid will automatically set the column width based on the content of the header and data cells in the column. This property allows you to change that to include only the headers or only the data.

Inherited From
`FlexGrid`
Type
`AutoSizeMode`

autoSizingColumnNg

Angular (EventEmitter) version of the Wijmo **autoSizingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizingColumn** Wijmo event name.

Type
EventEmitter

autoSizingRowNg

Angular (EventEmitter) version of the Wijmo **autoSizingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **autoSizingRow** Wijmo event name.

Type
EventEmitter

beginningEditNg

Angular (EventEmitter) version of the Wijmo **beginningEdit** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **beginningEdit** Wijmo event name.

Type
EventEmitter

bottomLeftCells

Gets the **`GridPanel`** that contains the bottom left cells.

The **`bottomLeftCells`** panel appears below the row headers, to the left of the **`columnFooters`** panel.

Inherited From
`FlexGrid`
Type
`GridPanel`

cellEditEndedNg

Angular (EventEmitter) version of the Wijmo **cellEditEnded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **cellEditEnded** Wijmo event name.

Type
EventEmitter

cellEditEndingNg

Angular (EventEmitter) version of the Wijmo **cellEditEnding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **cellEditEnding** Wijmo event name.

Type
EventEmitter

cellFactory

Gets or sets the **`CellFactory`** that creates and updates cells for this grid.

Inherited From
`FlexGrid`
Type
`CellFactory`

cells

Gets the **`GridPanel`** that contains the data cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

childItemsPath

Gets or sets the name of the property (or properties) used to generate child rows in hierarchical grids.

Set this property to a string to specify the name of the property that contains an item's child items (e.g. `'items'`).

If items at different levels child items with different names, then set this property to an array containing the names of the properties that contain child items et each level (e.g. `[ 'accounts', 'checks', 'earnings' ]`).

Example

Show me (http://jsfiddle.net/Wijmo5/t0ncmjwp)

Inherited From
`FlexGrid`
Type
any

clientSize

Gets the client size of the control (control size minus headers and scrollbars).

Inherited From
`FlexGrid`
Type
`Size`

cloneFrozenCells

Gets or sets a value that determines whether the FlexGrid should clone frozen cells and show then in a separate element to improve perceived performance while scrolling.

This property is set to null by default, which causes the grid to select the best setting depending on the browser.

Inherited From
`FlexGrid`
Type
boolean

collectionView

Gets the **ICollectionView** that contains the grid data.

Inherited From
FlexGrid
Type
ICollectionView

columnFooters

Gets the **GridPanel** that contains the column footer cells.

The **columnFooters** panel appears below the grid cells, to the right of the **bottomLeftCells** panel. It can be used to display summary information below the grid data.

The example below shows how you can add a row to the **columnFooters** panel to display summary data for columns that have the **aggregate** property set:

```
function addFooterRow(flex) {

  // create a GroupRow to show aggregates
  var row = new wijmo.grid.GroupRow();

  // add the row to the column footer panel
  flex.columnFooters.rows.push(row);

  // show a sigma on the header
  flex.bottomLeftCells.setCellData(0, 0, '\u03A3');
}
```

Inherited From
FlexGrid
Type
GridPanel

columnHeaders

Gets the **GridPanel** that contains the column header cells.

Inherited From
FlexGrid
Type
GridPanel

columnLayout

Gets or sets a JSON string that defines the current column layout.

The column layout string represents an array with the columns and their properties. It can be used to persist column layouts defined by users so they are preserved across sessions, and can also be used to implement undo/redo functionality in applications that allow users to modify the column layout.

The column layout string does not include **dataMap** properties, because data maps are not serializable.

Inherited From
`FlexGrid`
Type
string

columns

Gets the grid's column collection.

Inherited From
`FlexGrid`
Type
`ColumnCollection`

controlRect

Gets the bounding rectangle of the control in page coordinates.

Inherited From
`FlexGrid`
Type
`Rect`

STATIC controlTemplate

Overrides the template used to instantiate `FlexSheet` control.

Inherited From
`FlexSheet`
Type
any

copiedNg

Angular (EventEmitter) version of the Wijmo **copied** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **copied** Wijmo event name.

Type
EventEmitter

copyingNg

Angular (EventEmitter) version of the Wijmo **copying** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **copying** Wijmo event name.

Type
EventEmitter

deferResizing

Gets or sets a value that determines whether row and column resizing should be deferred until the user releases the mouse button.

By default, **`deferResizing`** is set to false, causing rows and columns to be resized as the user drags the mouse. Setting this property to true causes the grid to show a resizing marker and to resize the row or column only when the user releases the mouse button.

Inherited From
`FlexGrid`
Type
boolean

definedNames

Gets an array the **`IDefinedName`** objects representing named ranges/expressions defined in the **FlexSheet**.

Inherited From
`FlexSheet`
Type
`ObservableArray`

deletedRowNg

Angular (EventEmitter) version of the Wijmo **deletedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **deletedRow** Wijmo event name.

Type
EventEmitter

deletingRowNg

Angular (EventEmitter) version of the Wijmo **deletingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **deletingRow** Wijmo event name.

Type
EventEmitter

draggedColumnNg

Angular (EventEmitter) version of the Wijmo **draggedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggedColumn** Wijmo event name.

Type
EventEmitter

draggedRowNg

Angular (EventEmitter) version of the Wijmo **draggedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggedRow** Wijmo event name.

Type
EventEmitter

draggingColumnNg

Angular (EventEmitter) version of the Wijmo **draggingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingColumn** Wijmo event name.

Type
EventEmitter

draggingColumnOverNg

Angular (EventEmitter) version of the Wijmo **draggingColumnOver** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingColumnOver** Wijmo event name.

Type
EventEmitter

draggingRowColumnNg

Angular (EventEmitter) version of the Wijmo **draggingRowColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRowColumn** Wijmo event name.

Type
EventEmitter

draggingRowNg

Angular (EventEmitter) version of the Wijmo **draggingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRow** Wijmo event name.

Type
EventEmitter

draggingRowOverNg

Angular (EventEmitter) version of the Wijmo **draggingRowOver** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **draggingRowOver** Wijmo event name.

Type
EventEmitter

droppingRowColumnNg

Angular (EventEmitter) version of the Wijmo **droppingRowColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **droppingRowColumn** Wijmo event name.

Type
EventEmitter

editableCollectionView

Gets the **IEditableCollectionView** that contains the grid data.

Inherited From
FlexGrid
Type
IEditableCollectionView

editRange

Gets a **CellRange** that identifies the cell currently being edited.

Inherited From
FlexGrid
Type
CellRange

formatItemNg

Angular (EventEmitter) version of the Wijmo **formatItem** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **formatItem** Wijmo event name.

Type
EventEmitter

frozenColumns

Gets or sets the number of frozen columns.

Frozen columns do not scroll horizontally, but the cells they contain may be selected and edited.

Inherited From
FlexGrid
Type
number

frozenRows

Gets or sets the number of frozen rows.

Frozen rows do not scroll vertically, but the cells they contain may be selected and edited.

Inherited From
`FlexGrid`
Type
number

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

groupCollapsedChangedNg

Angular (EventEmitter) version of the Wijmo **groupCollapsedChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **groupCollapsedChanged** Wijmo event name.

Type
EventEmitter

groupCollapsedChangingNg

Angular (EventEmitter) version of the Wijmo **groupCollapsedChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **groupCollapsedChanging** Wijmo event name.

Type
EventEmitter

groupHeaderFormat

Gets or sets the format string used to create the group header content.

The string may contain any text, plus the following replacement strings:

- **{name}**: The name of the property being grouped on.
- **{value}**: The value of the property being grouped on.
- **{level}**: The group level.
- **{count}**: The total number of items in this group.

If a column is bound to the grouping property, the column header is used to replace the `{name}` parameter, and the column's format and data maps are used to calculate the `{value}` parameter. If no column is available, the group information is used instead.

You may add invisible columns bound to the group properties in order to customize the formatting of the group header cells.

The default value for this property is
`'{name}: <b>{value}</b>({count:n0} items)'`, which creates group headers similar to
`'Country: UK (12 items)'` or
`'Country: Japan (8 items)'`.

Inherited From
`FlexGrid`
Type
string

headersVisibility

Gets or sets a value that determines whether the row and column headers are visible.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

imeEnabled

Gets or sets a value that determines whether the grid should support Input Method Editors (IME) while not in edit mode.

This property is relevant only for sites/applications in Japanese, Chinese, Korean, and other languages that require IME support.

Inherited From
`FlexGrid`
Type
boolean

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isFunctionListOpen

Gets a value indicating whether the function list is opened.

Inherited From
`FlexSheet`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isReadOnly

Gets or sets a value that determines whether the user can modify cell values using the mouse and keyboard.

Inherited From
`FlexGrid`
Type
boolean

isTabHolderVisible

Gets or sets a value indicating whether the TabHolder is visible.

Inherited From
`FlexSheet`
Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets a formatter function used to customize cells on this grid.

The formatter function can add any content to any cell. It provides complete flexibility over the appearance and behavior of grid cells.

If specified, the function should take four parameters: the **GridPanel** that contains the cell, the row and column indices of the cell, and the HTML element that represents the cell. The function will typically change the **innerHTML** property of the cell element.

For example:

```
flex.itemFormatter = function(panel, r, c, cell) {
  if (panel.cellType == CellType.Cell) {

    // draw sparklines in the cell
    var col = panel.columns[c];
    if (col.name == 'sparklines') {
      cell.innerHTML = getSparklike(panel, r, c);
    }
  }
}
```

Note that the FlexGrid recycles cells, so if your **itemFormatter** modifies the cell's style attributes, you must make sure that it resets these attributes for cells that should not have them. For example:

```
flex.itemFormatter = function(panel, r, c, cell) {

  // reset attributes we are about to customize
  var s = cell.style;
  s.color = '';
  s.backgroundColor = '';

  // customize color and backgroundColor attributes for this cell
  ...
}
```

If you have a scenario where multiple clients may want to customize the grid rendering (for example when creating directives or re-usable libraries), consider using the **formatItem** event instead. The event allows multiple clients to attach their own handlers.

Inherited From
FlexGrid
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** that contains items shown on the grid.

Inherited From
`FlexGrid`
Type
any

itemsSourceChangedNg

Angular (EventEmitter) version of the Wijmo **itemsSourceChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **itemsSourceChanged** Wijmo event name.

Type
EventEmitter

itemValidator

Gets or sets a validator function to determine whether cells contain valid data.

If specified, the validator function should take two parameters containing the cell's row and column indices, and should return a string containing the error description.

This property is especially useful when dealing with unbound grids, since bound grids can be validated using the **`getError`** property instead.

This example shows how you could prevent cells from containing the same data as the cell immediately above it:

```
// check that the cell above doesn't contain the same value as this one
theGrid.itemValidator = function (row, col) {
  if (row > 0) {
    var valThis = theGrid.getCellData(row, col, false),
        valPrev = theGrid.getCellData(row - 1, col, false);
    if (valThis != null && valThis == valPrev) {
      return 'This is a duplicate value...'
    }
  }
  return null; // no errors
}
```

Inherited From
`FlexGrid`
Type
Function

keyActionEnter

Gets or sets the action to perform when the ENTER key is pressed.

The default setting for this property is **MoveDown**, which causes the control to move the selection to the next row. This is the standard Excel behavior.

Inherited From
FlexGrid
Type
KeyAction

keyActionTab

Gets or sets the action to perform when the TAB key is pressed.

The default setting for this property is **None**, which causes the browser to select the next or previous controls on the page when the TAB key is pressed. This is the recommended setting to improve page accessibility.

In previous versions, the default was set to **Cycle**, which caused the control to move the selection across and down the grid. This is the standard Excel behavior, but is not good for accessibility.

There is also a **CycleOut** setting that causes the selection to move through the cells (as **Cycle**), and then on to the next/previous control on the page when the last or first cells are selected.

Inherited From
FlexGrid
Type
KeyAction

loadedNg

Angular (EventEmitter) version of the Wijmo **loaded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loaded** Wijmo event name.

Type
EventEmitter

loadedRowsNg

Angular (EventEmitter) version of the Wijmo **loadedRows** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loadedRows** Wijmo event name.

Type
EventEmitter

loadingRowsNg

Angular (EventEmitter) version of the Wijmo **loadingRows** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **loadingRows** Wijmo event name.

Type
EventEmitter

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

mergeManager

Gets or sets the **`MergeManager`** object responsible for determining how cells should be merged.

Inherited From
`FlexGrid`
Type
`MergeManager`

newRowAtTop

Gets or sets a value that indicates whether the new row template should be located at the top of the grid or at the bottom.

If you set the **`newRowAtTop`** property to true, and you want the new row template to remain visible at all times, set the **`frozenRows`** property to one. This will freeze the new row template at the top so it won't scroll off the view.

The new row template will be displayed only if the **`allowAddNew`** property is set to true and if the **`itemsSource`** object supports adding new items.

Inherited From
`FlexGrid`
Type
boolean

pastedCellNg

Angular (EventEmitter) version of the Wijmo **pastedCell** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pastedCell** Wijmo event name.

Type
EventEmitter

pastedNg

Angular (EventEmitter) version of the Wijmo **pasted** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pasted** Wijmo event name.

Type
EventEmitter

pastingCellNg

Angular (EventEmitter) version of the Wijmo **pastingCell** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pastingCell** Wijmo event name.

Type
EventEmitter

pastingNg

Angular (EventEmitter) version of the Wijmo **pasting** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **pasting** Wijmo event name.

Type
EventEmitter

prepareCellForEditNg

Angular (EventEmitter) version of the Wijmo **prepareCellForEdit** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **prepareCellForEdit** Wijmo event name.

Type
EventEmitter

preserveOutlineState

Gets or sets a value that determines whether the grid should preserve the expanded/collapsed state of nodes when the data is refreshed.

The **`preserveOutlineState`** property implementation is based on JavaScript's **Map** object, which is not available in IE 9 or 10.

Inherited From
`FlexGrid`
Type
boolean

preserveSelectedState

Gets or sets a value that determines whether the grid should preserve the selected state of rows when the data is refreshed.

Inherited From
`FlexGrid`
Type
boolean

quickAutoSize

Gets or sets a value that determines whether the grid should optimize performance over precision when auto-sizing columns.

Setting this property to false disables quick auto-sizing. Setting it to true enables the feature, subject to the value of each column's **`quickAutoSize`** property. Setting it to null (the default value) enables the feature for grids that don't have a custom **`itemFormatter`** or handlers attached to the **`formatItem`** event.

Inherited From
`FlexGrid`
Type
boolean

resizedColumnNg

Angular (EventEmitter) version of the Wijmo **resizedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizedColumn** Wijmo event name.

Type
EventEmitter

resizedRowNg

Angular (EventEmitter) version of the Wijmo **resizedRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizedRow** Wijmo event name.

Type
EventEmitter

resizingColumnNg

Angular (EventEmitter) version of the Wijmo **resizingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizingColumn** Wijmo event name.

Type
EventEmitter

resizingRowNg

Angular (EventEmitter) version of the Wijmo **resizingRow** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **resizingRow** Wijmo event name.

Type
EventEmitter

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rowAddedNg

Angular (EventEmitter) version of the Wijmo **rowAdded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowAdded** Wijmo event name.

Type
EventEmitter

rowEditEndedNg

Angular (EventEmitter) version of the Wijmo **rowEditEnded** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditEnded** Wijmo event name.

Type
EventEmitter

rowEditEndingNg

Angular (EventEmitter) version of the Wijmo **rowEditEnding** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditEnding** Wijmo event name.

Type
EventEmitter

rowEditStartedNg

Angular (EventEmitter) version of the Wijmo **rowEditStarted** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditStarted** Wijmo event name.

Type
EventEmitter

rowEditStartingNg

Angular (EventEmitter) version of the Wijmo **rowEditStarting** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rowEditStarting** Wijmo event name.

Type
EventEmitter

rowHeaderPath

Gets or sets the name of the property used to create row header cells.

Row header cells are not visible or selectable. They are meant for use with accessibility tools.

Inherited From
`FlexGrid`
Type
string

rowHeaders

Gets the `GridPanel` that contains the row header cells.

Inherited From
`FlexGrid`
Type
`GridPanel`

rows

Gets the grid's row collection.

Inherited From
`FlexGrid`
Type
`RowCollection`

scrollPosition

Gets or sets a `Point` that represents the value of the grid's scrollbars.

Inherited From
`FlexGrid`
Type
`Point`

scrollPositionChangedNg

Angular (EventEmitter) version of the Wijmo **scrollPositionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **scrollPositionChanged** Wijmo event name.

Type
EventEmitter

scrollSize

Gets the size of the grid content in pixels.

Inherited From
`FlexGrid`
Type
`Size`

selectedItems

Gets or sets an array containing the data items that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedRows

Gets or sets an array containing the rows that are currently selected.

Note: this property can be read in all selection modes, but it can be set only when `selectionMode` is set to **SelectionMode.ListBox**.

Inherited From
`FlexGrid`
Type
any[]

selectedSheet

Gets the current **Sheet** in the **FlexSheet**.

Inherited From
FlexSheet
Type
Sheet

selectedSheetChangedNg

Angular (EventEmitter) version of the Wijmo **selectedSheetChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectedSheetChanged** Wijmo event name.

Type
EventEmitter

selectedSheetIndex

Gets or sets the index of the current sheet in the **FlexSheet**.

Inherited From
FlexSheet
Type
number

selection

Gets or sets the current selection.

Inherited From
FlexGrid
Type
CellRange

selectionChangedNg

Angular (EventEmitter) version of the Wijmo **selectionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanged** Wijmo event name.

Type
EventEmitter

selectionChangingNg

Angular (EventEmitter) version of the Wijmo **selectionChanging** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanging** Wijmo event name.

Type
EventEmitter

selectionMode

Gets or sets the current selection mode.

Inherited From
`FlexGrid`
Type
`SelectionMode`

sheetClearedNg

Angular (EventEmitter) version of the Wijmo **sheetCleared** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sheetCleared** Wijmo event name.

Type
EventEmitter

sheets

Gets the collection of **Sheet** objects representing workbook sheets.

Inherited From
`FlexSheet`
Type
`SheetCollection`

showAlternatingRows

Gets or sets a value that determines whether the grid should add the 'wj-alt' class to cells in alternating rows.

Setting this property to false disables alternate row styles without any changes to the CSS.

Inherited From
`FlexGrid`
Type
boolean

showDropDown

Gets or sets a value that indicates whether the grid adds drop-down buttons to the cells in columns that have the **`showDropDown`** property set to true.

The drop-down buttons are shown only on columns that have a **dataMap** set and are editable. Clicking on the drop-down buttons causes the grid to show a list where users can select the value for the cell.

Cell drop-downs require the wijmo.input module to be loaded.

Inherited From
`FlexGrid`
Type
boolean

showErrors

Gets or sets a value that determines whether the grid should add the 'wj-state-invalid' class to cells that contain validation errors, and tooltips with error descriptions.

The grid detects validation errors using the **`itemValidator`** property or the **`getError`** property on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showFilterIcons

Gets or sets the visiblity of the filter icon.

Inherited From
`FlexSheet`
Type
boolean

showGroups

Gets or sets a value that determines whether the grid should insert group rows to delimit data groups.

Data groups are created by modifying the **`groupDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

showMarquee

Gets or sets a value that indicates whether the grid should display a marquee element around the current selection.

Inherited From
`FlexGrid`
Type
boolean

showSelectedHeaders

Gets or sets a value that indicates whether the grid should add class names to indicate selected header cells.

Inherited From
`FlexGrid`
Type
`HeadersVisibility`

showSort

Gets or sets a value that determines whether the grid should display sort indicators in the column headers.

Sorting is controlled by the **`sortDescriptions`** property of the **`ICollectionView`** object used as a the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

sortedColumnNg

Angular (EventEmitter) version of the Wijmo **sortedColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sortedColumn** Wijmo event name.

Type
EventEmitter

sortingColumnNg

Angular (EventEmitter) version of the Wijmo **sortingColumn** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **sortingColumn** Wijmo event name.

Type
EventEmitter

sortManager

Gets the **`SortManager`** instance that controls **FlexSheet** sorting.

Inherited From
`FlexSheet`
Type
`SortManager`

sortRowIndex

Gets or sets the index of row in the column header panel that shows and changes the current sort.

This property is set to null by default, causing the last row in the **`columnHeaders`** panel to act as the sort row.

Inherited From
`FlexGrid`
Type
number

stickyHeaders

Gets or sets a value that determines whether column headers should remain visible when the user scrolls the document.

Inherited From
`FlexGrid`
Type
boolean

topLeftCells

Gets the **`GridPanel`** that contains the top left cells (to the left of the column headers).

Inherited From
`FlexGrid`
Type
`GridPanel`

treeIndent

Gets or sets the indent used to offset row groups of different levels.

Inherited From
`FlexGrid`
Type
number

undoStack

Gets the **UndoStack** instance that controls undo and redo operations of the **FlexSheet**.

Inherited From
FlexSheet
Type
UndoStack

unknownFunctionNg

Angular (EventEmitter) version of the Wijmo **unknownFunction** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **unknownFunction** Wijmo event name.

Type
EventEmitter

updatedLayoutNg

Angular (EventEmitter) version of the Wijmo **updatedLayout** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatedLayout** Wijmo event name.

Type
EventEmitter

updatedViewNg

Angular (EventEmitter) version of the Wijmo **updatedView** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatedView** Wijmo event name.

Type
EventEmitter

updatingLayoutNg

Angular (EventEmitter) version of the Wijmo **updatingLayout** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatingLayout** Wijmo event name.

Type
EventEmitter

updatingViewNg

Angular (EventEmitter) version of the Wijmo **updatingView** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **updatingView** Wijmo event name.

Type
EventEmitter

validateEdits

Gets or sets a value that determines whether the grid should remain in edit mode when the user tries to commit edits that fail validation.

The grid detects validation errors by calling the **`getError`** method on the grid's **`itemsSource`**.

Inherited From
`FlexGrid`
Type
boolean

viewRange

Gets the range of cells currently in view.

Inherited From
`FlexGrid`
Type
`CellRange`

virtualizationThreshold

Gets or sets the minimum number of rows required to enable virtualization.

This property is set to zero by default, meaning virtualization is always enabled. This improves binding performance and memory requirements, at the expense of a small performance decrease while scrolling.

If your grid has a small number of rows (about 50 to 100), you may be able to improve scrolling performance by setting this property to a slightly higher value (like 150). This will disable virtualization and will slow down binding, but may improve perceived scroll performance.

Setting this property to values higher than 200 is not recommended. Loading times will become too long; the grid will freeze for a few seconds while creating cells for all rows, and the browser will become slow because of the large number of elements on the page.

Inherited From
`FlexGrid`
Type
number

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addBoundSheet

```
addBoundSheet(sheetName: string, source: any, pos?: number, grid?: FlexGrid): Sheet
```

Add a bound **Sheet** to the **FlexSheet**.

Parameters

- **sheetName**: **string**
  The name of the **Sheet**.

- **source**: **any**
  The items source for the **Sheet**.

- **pos**: **number** OPTIONAL
  The position in the **sheets** collection.

- **grid**: **FlexGrid** OPTIONAL
  The **FlexGrid** instance associated with the **Sheet**. If not specified then new **FlexGrid** instance will be created.

Inherited From
FlexSheet
Returns
Sheet

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

addFunction

```
addFunction(name: string, func: Function, description?: string, minParamsCount?: number, maxParamsCount?: number): void
```

Add custom function in **FlexSheet**.

Parameters

- **name**: **string**
  the name of the custom function.

- **func**: **Function**
  the custom function.
  The function signature looks as follows:

  ```
  function (...params: any[][][]): any;
  ```

  The function takes a variable number of parameters, each parameter corresponds to an expression passed as a function argument. Independently of whether the expression passed as a function argument resolves to a single value or to a cell range, each parameter value is always a two dimensional array of resolved values. The number of rows (first index) and columns (second index) in the array corresponds to the size of the specified cell range. In case where argument is an expression that resolves to a single value, it will be a one-to-one array where its value can be retrieved using the param[0][0] indexer.
  The sample below adds a custom Sum Product function ('customSumProduct') to the FlexSheet:

  ```
  flexSheet.addFunction('customSumProduct', (...params: any[][][]) => {
     let result = 0,
         range1 = params[0],
         range2 = params[1];

     if (range1.length > 0 && range1.length === range2.length && range1[0].length === range2[0].length) {
         for (let i = 0; i < range1.length; i++) {
             for (let j = 0; j < range1[0].length; j++) {
                 result += range1[i][j] * range2[i][j];
             }
         }
     }
     return result;
  }, 'Custom SumProduct Function', 2, 2);
  ```

  After adding this function, it can be used it in sheet cell expressions, like here:

  ```
  =customSumProduct(A1:B5, B1:C5)
  ```

- **description**: **string** OPTIONAL
  the description of the custom function, it will be shown in the function autocompletion of the **FlexSheet**.

- **minParamsCount**: **number** OPTIONAL

the minimum count of the parameter that the function need.

- **maxParamsCount**: **number** OPTIONAL
  the maximum count of the parameter that the function need. If the count of the parameters in the custom function is arbitrary, the minParamsCount and maxParamsCount should be set to null.

Inherited From
FlexSheet
Returns
void

addUnboundSheet

```
addUnboundSheet(sheetName?: string, rows?: number, cols?: number, pos?: number, grid?: FlexGrid): Sheet
```

Add an unbound **Sheet** to the **FlexSheet**.

Parameters

- **sheetName**: **string** OPTIONAL
  The name of the Sheet.
- **rows**: **number** OPTIONAL
  The row count of the Sheet.
- **cols**: **number** OPTIONAL
  The column count of the Sheet.
- **pos**: **number** OPTIONAL
  The position in the **sheets** collection.
- **grid**: **FlexGrid** OPTIONAL
  The **FlexGrid** instance associated with the **Sheet**. If not specified then new **FlexGrid** instance will be created.

Inherited From
FlexSheet
Returns
Sheet

applyCellsStyle

```
applyCellsStyle(cellStyle: ICellStyle, cells?: CellRange[], isPreview?: boolean): void
```

Apply the style to a range of cells.

Parameters

- **cellStyle**: **ICellStyle**
  The **ICellStyle** object to apply.
- **cells**: **CellRange**[] OPTIONAL
  An array of **CellRange** objects to apply the style to. If not specified then style is applied to the currently selected cells.
- **isPreview**: **boolean** OPTIONAL
  Indicates whether the applied style is just for preview.

Inherited From
FlexSheet
Returns
void

applyFunctionToCell

```
applyFunctionToCell(): void
```

Inserts the selected function from the function list to the cell value editor.

Inherited From
FlexSheet
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

autoSizeColumn

```
autoSizeColumn(c: number, header?: boolean, extra?: number): void
```

Resizes a column to fit its content.

Parameters

- **c**: **number**
  Index of the column to resize.

- **header**: **boolean** OPTIONAL
  Whether the column index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeColumns

```
autoSizeColumns(firstColumn?: number, lastColumn?: number, header?: boolean, extra?: number): void
```

Resizes a range of columns to fit their content.

The grid will always measure all rows in the current view range, plus up to 2,000 rows not currently in view. If the grid contains a large amount of data (say 50,000 rows), then not all rows will be measured since that could potentially take a long time.

Parameters

- **firstColumn**: **number** OPTIONAL
  Index of the first column to resize (defaults to the first column).
- **lastColumn**: **number** OPTIONAL
  Index of the last column to resize (defaults to the last column).
- **header**: **boolean** OPTIONAL
  Whether the column indices refer to regular or header columns.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRow

```
autoSizeRow(r: number, header?: boolean, extra?: number): void
```

Resizes a row to fit its content.

Parameters

- **r**: **number**
  Index of the row to resize.

- **header**: **boolean** OPTIONAL
  Whether the row index refers to a regular or a header row.

- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

autoSizeRows

```
autoSizeRows(firstRow?: number, lastRow?: number, header?: boolean, extra?: number): void
```

Resizes a range of rows to fit their content.

Parameters

- **firstRow**: **number** OPTIONAL
  Index of the first row to resize.
- **lastRow**: **number** OPTIONAL
  Index of the last row to resize.
- **header**: **boolean** OPTIONAL
  Whether the row indices refer to regular or header rows.
- **extra**: **number** OPTIONAL
  Extra spacing, in pixels.

Inherited From
`FlexGrid`
Returns
void

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **`endUpdate`**.

Inherited From
`Control`
Returns
void

canEditCell

```
canEditCell(r: number, c: number): void
```

Gets a value that indicates whether a given cell can be edited.

Parameters

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

Inherited From
FlexGrid
Returns
void

clear

```
clear(): void
```

Clears the content of the **FlexSheet** control.

Inherited From
FlexSheet
Returns
void

collapseGroupsToLevel

```
collapseGroupsToLevel(level: number): void
```

Collapses all the group rows to a given level.

Parameters

- **level**: **number**
  Maximum group level to show.

Inherited From
FlexGrid
Returns
void

containsFocus

```
containsFocus(): boolean
```

Overrides the base class method to take into account the function list.

Inherited From
FlexSheet
Returns
boolean

STATIC convertNumberToAlpha

```
convertNumberToAlpha(c: number): string
```

Converts the number value to its corresponding alpha value. For instance: 0, 1, 2...to a, b, c...

Parameters

- **c**: **number**
  The number value need to be converted.

Inherited From
FlexSheet
Returns
string

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **beginUpdate/endUpdate** block.

The control will not be updated until the function has been executed. This method ensures **endUpdate** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
Control
Returns
void

deleteColumns

```
deleteColumns(index?: number, count?: number): void
```

Deletes columns from the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The starting index of the deleting columns. If not specified then columns will be deleted starting from the first column of the current selection.

- **count**: **number** OPTIONAL
  The numbers of columns to delete. If not specified then one column will be deleted.

Inherited From
FlexSheet
Returns
void

deleteRows

```
deleteRows(index?: number, count?: number): void
```

Deletes rows from the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The starting index of the deleting rows. If not specified then rows will be deleted starting from the first row of the current selection.

- **count**: **number** OPTIONAL
  The numbers of rows to delete. If not specified then one row will be deleted.

Inherited From
FlexSheet
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

Inherited From
`FlexSheet`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
`Control`
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **`beginUpdate`**.

Inherited From
`Control`
Returns
void

evaluate

```
evaluate(formula: string, format?: string, sheet?: Sheet): any
```

Evaluates a formula.

FlexSheet formulas follow the Excel syntax, including a large subset of the functions supported by Excel. A complete list of the functions supported by **FlexSheet** can be found here: **FlexSheet Functions**.

Parameters

- **formula**: **string**
  The formula to evaluate. The formula may start with an optional equals sign ('=').
- **format**: **string** OPTIONAL
  If specified, defines the .Net format that will be applied to the evaluated value.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose data will be used for evaluation. If not specified then the current sheet is used.

Inherited From
FlexSheet
Returns
any

finishEditing

```
finishEditing(cancel?: boolean): boolean
```

Commits any pending edits and exits edit mode.

Parameters

- **cancel**: **boolean** OPTIONAL
  Whether pending edits should be canceled or committed.

Inherited From
FlexGrid
Returns
boolean

focus

```
focus(): void
```

Overridden to set the focus to the grid without scrolling the whole grid into view.

Inherited From
`FlexGrid`
Returns
void

freezeAtCursor

```
freezeAtCursor(): void
```

Freeze or unfreeze the columns and rows of the **FlexSheet** control.

Inherited From
`FlexSheet`
Returns
void

getCellBoundingRect

```
getCellBoundingRect(r: number, c: number, raw?: boolean): Rect
```

Gets a the bounds of a cell element in viewport coordinates.

This method returns the bounds of cells in the **`cells`** panel (scrollable data cells). To get the bounds of cells in other panels, use the **`getCellBoundingRect`** method in the appropriate **`GridPanel`** object.

The returned value is a **`Rect`** object which contains the position and dimensions of the cell in viewport coordinates. The viewport coordinates are the same used by the **getBoundingClientRect** method.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **raw**: **boolean** OPTIONAL
  Whether to return the rectangle in raw panel coordinates as opposed to viewport coordinates.

Inherited From
`FlexGrid`
Returns
`Rect`

getCellData

```
getCellData(r: number, c: number, formatted: boolean): any
```

Gets the value stored in a cell in the scrollable area of the grid.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **number**
  Index of the column that contains the cell.
- **formatted**: **boolean**
  Whether to format the value for display.

Inherited From
`FlexGrid`
Returns
any

getCellValue

```
getCellValue(rowIndex: number, colIndex: number, formatted?: boolean, sheet?: Sheet): any
```

Gets the evaluated cell value.

Unlike the **getCellData** method that returns a raw data that can be a value or a formula, the **getCellValue** method always returns an evaluated value, that is if the cell contains a formula then it will be evaluated first and the resulting value will be returned.

Parameters

- **rowIndex**: **number**
  The row index of the cell.
- **colIndex**: **number**
  The column index of the cell.
- **formatted**: **boolean** OPTIONAL
  Indicates whether to return an original or a formatted value of the cell.
- **sheet**: **Sheet** OPTIONAL
  The **Sheet** whose value to evaluate. If not specified then the data from current sheet is used.

Inherited From
FlexSheet
Returns
any

getClipString

```
getClipString(rng?: CellRange): string
```

Gets the content of a **CellRange** as a string suitable for copying to the clipboard.

FlexSheet overrides this method to support multiple rows or columns selection in **FlexSheet**.

Hidden rows and columns are not included in the clip string.

Parameters

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexSheet
Returns
string

getColumn

```
getColumn(name: string): Column
```

Gets a column by name or by binding.

The method searches the column by name. If a column with the given name is not found, it searches by binding. The searches are case-sensitive.

Parameters

- **name**: **string**
  The name or binding to find.

Inherited From
FlexGrid
Returns
Column

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
Control
Returns
Control

getMergedRange

```
getMergedRange(panel: GridPanel, r: number, c: number, clip?: boolean): CellRange
```

Gets a **CellRange** that specifies the merged extent of a cell in a **GridPanel**. This method overrides the getMergedRange method of its parent class FlexGrid

Parameters

- **panel**: **GridPanel**
  GridPanel that contains the range.

- **r**: **number**
  Index of the row that contains the cell.

- **c**: **number**
  Index of the column that contains the cell.

- **clip**: **boolean** OPTIONAL
  Whether to clip the merged range to the grid's current view range.

Inherited From
FlexSheet
Returns
CellRange

getSelectedState

```
getSelectedState(r: number, c: number): SelectedState
```

Gets a **SelectedState** value that indicates the selected state of a cell.

Parameters

- **r**: **number**
  Row index of the cell to inspect.

- **c**: **number**
  Column index of the cell to inspect.

Inherited From
FlexGrid
Returns
SelectedState

getSelectionFormatState

```
getSelectionFormatState(): IFormatState
```

Gets the **IFormatState** object describing formatting of the selected cells.

Inherited From
FlexSheet
Returns
IFormatState

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **ComboBox** control, it will override the template defined by the **DropDown** base class.

Inherited From
Control
Returns
string

hideFunctionList

```
hideFunctionList(): void
```

Close the function list.

Inherited From
FlexSheet
Returns
void

hitTest

```
hitTest(pt: any, y?: any): HitTestInfo
```

Gets a **HitTestInfo** object with information about a given point.

For example:

```
// hit test a point when the user clicks on the grid
flex.hostElement.addEventListener('click', function (e) {
  var ht = flex.hitTest(e.pageX, e.pageY);
  console.log('you clicked a cell of type "' +
    wijmo.grid.CellType[ht.cellType] + '".');
});
```

Parameters

- **pt**: **any**
  Point to investigate, in page coordinates, or a MouseEvent object, or x coordinate of the point.

- **y**: **any** OPTIONAL
  Y coordinate of the point in page coordinates (if the first parameter is a number).

Inherited From
FlexGrid
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

insertColumns

```
insertColumns(index?: number, count?: number): void
```

Inserts columns in the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The position where new columns should be added. If not specified then columns will be added before the left column of the current selection.

- **count**: **number** OPTIONAL
  The numbers of columns to add. If not specified then one column will be added.

Inherited From
FlexSheet
Returns
void

insertRows

```
insertRows(index?: number, count?: number): void
```

Inserts rows in the current **Sheet** of the **FlexSheet** control.

Parameters

- **index**: **number** OPTIONAL
  The position where new rows should be added. If not specified then rows will be added before the first row of the current selection.

- **count**: **number** OPTIONAL
  The numbers of rows to add. If not specified then one row will be added.

Inherited From
FlexSheet
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

isRangeValid

```
isRangeValid(rng: CellRange): boolean
```

Checks whether a given CellRange is valid for this grid's row and column collections.

Parameters

- **rng**: **CellRange**
  Range to check.

Inherited From
`FlexGrid`
Returns
boolean

load

```
load(workbook: any): void
```

Loads the workbook into **FlexSheet**. This method works with JSZip 2.5.

For example:

```
// This sample opens an xlsx file chosen through Open File
// dialog and fills FlexSheet


// HTML
<input type="file"
    id="importFile"
    accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
/>
<div id="flexHost"></>


// JavaScript
var flexSheet = new wijmo.grid.FlexSheet("#flexHost"),
    importFile = document.getElementById('importFile');

importFile.addEventListener('change', function () {
    loadWorkbook();
});

function loadWorkbook() {
    var reader,
        file = importFile.files[0];
    if (file) {
        reader = new FileReader();
        reader.onload = function (e) {
            flexSheet.load(reader.result);
        };
        reader.readAsArrayBuffer(file);
    }
}
```

Parameters

- **workbook**: **any**

  A workbook instance or a Blob instance or a base-64 string or an ArrayBuffer containing xlsx file content.

Inherited From
FlexSheet
Returns
void

loadAsync

```
loadAsync(workbook: any, onLoaded?: (workbook: wijmo.xlsx.Workbook), onError?: (reason?: any)): void
```

Loads the workbook into **FlexSheet** asynchronously. This method works with JSZip 3.0.

Parameters

- **workbook**: **any**
  A workbook instance or a Blob instance or a base-64 string or an ArrayBuffer containing xlsx file content.

- **onLoaded**: **(workbook: wijmo.xlsx.Workbook)** OPTIONAL
  This callback provides an approach to get the loaded workbook instance. Since this method is an asynchronous method, user is not able to get the loaded workbook instance immediately. User has to get the loaded workbook instance through this callback. This has a single parameter, the loaded workbook instance. It is passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when loading. This has a single parameter, the failure reason. The return value is passed to user if he wants to catch the load failure reason.

  For example:

```
flexsheet.loadAsync(blob, function (workbook) {

     // user can access the loaded workbook instance in this callback.
     var app = worksheet.application ;
     ...
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of load failure is ' + reason);
});
```

Inherited From
`FlexSheet`
Returns
void

mergeRange

```
mergeRange(cells?: CellRange, isCopyMergeCell?: boolean): void
```

Merges the selected **CellRange** into one cell.

Parameters

- **cells**: **CellRange** OPTIONAL
  The **CellRange** to merge.

- **isCopyMergeCell**: **boolean** OPTIONAL
  This parameter indicates that merge operation is done by copy\paste merge cell or not.

Inherited From
FlexSheet
Returns
void

onAutoSizedColumn

```
onAutoSizedColumn(e: CellRangeEventArgs): void
```

Raises the **autoSizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizedRow

```
onAutoSizedRow(e: CellRangeEventArgs): void
```

Raises the **autoSizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onAutoSizingColumn

```
onAutoSizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onAutoSizingRow

```
onAutoSizingRow(e: CellRangeEventArgs): boolean
```

Raises the **autoSizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onBeginningEdit

```
onBeginningEdit(e: CellRangeEventArgs): boolean
```

Raises the **beginningEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onCellEditEnded

```
onCellEditEnded(e: CellRangeEventArgs): void
```

Raises the **cellEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCellEditEnding

```
onCellEditEnding(e: CellEditEndingEventArgs): boolean
```

Raises the **cellEditEnding** event.

Parameters

- **e**: **CellEditEndingEventArgs**
  CellEditEndingEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onColumnChanged

```
onColumnChanged(e: RowColumnChangedEventArgs): void
```

Raises the columnChanged event.

Parameters

- **e**: **RowColumnChangedEventArgs**

Inherited From
FlexSheet
Returns
void

onCopied

```
onCopied(e: CellRangeEventArgs): void
```

Raises the **copied** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onCopying

```
onCopying(e: CellRangeEventArgs): boolean
```

Raises the **copying** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDeletedRow

```
onDeletedRow(e: CellRangeEventArgs): void
```

Raises the **deletedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDeletingRow

```
onDeletingRow(e: CellRangeEventArgs): boolean
```

Raises the **deletingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggedColumn

```
onDraggedColumn(e: CellRangeEventArgs): void
```

Raises the **draggedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggedRow

```
onDraggedRow(e: CellRangeEventArgs): void
```

Raises the **draggedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onDraggingColumn

```
onDraggingColumn(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingColumnOver

```
onDraggingColumnOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingColumnOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRow

```
onDraggingRow(e: CellRangeEventArgs): boolean
```

Raises the **draggingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDraggingRowColumn

```
onDraggingRowColumn(e: DraggingRowColumnEventArgs): void
```

Raises the draggingRowColumn event.

Parameters

- **e**: **DraggingRowColumnEventArgs**

Inherited From
FlexSheet
Returns
void

onDraggingRowOver

```
onDraggingRowOver(e: CellRangeEventArgs): boolean
```

Raises the **draggingRowOver** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onDroppingRowColumn

```
onDroppingRowColumn(e?: EventArgs): void
```

Raises the droppingRowColumn event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexSheet
Returns
void

onFormatItem

```
onFormatItem(e: FormatItemEventArgs): void
```

Raises the **formatItem** event.

Parameters

- **e**: **FormatItemEventArgs**
  FormatItemEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onGroupCollapsedChanged

```
onGroupCollapsedChanged(e: CellRangeEventArgs): void
```

Raises the **groupCollapsedChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onGroupCollapsedChanging

```
onGroupCollapsedChanging(e: CellRangeEventArgs): boolean
```

Raises the **groupCollapsedChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onItemsSourceChanged

```
onItemsSourceChanged(e?: EventArgs): void
```

Raises the **itemsSourceChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoaded

```
onLoaded(e?: EventArgs): void
```

Raises the loaded event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexSheet
Returns
void

onLoadedRows

```
onLoadedRows(e?: EventArgs): void
```

Raises the **loadedRows** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onLoadingRows

```
onLoadingRows(e: CancelEventArgs): boolean
```

Raises the **loadingRows** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onPasted

```
onPasted(e: CellRangeEventArgs): void
```

Raises the **pasted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPastedCell

```
onPastedCell(e: CellRangeEventArgs): void
```

Raises the **pastedCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPasting

```
onPasting(e: CellRangeEventArgs): boolean
```

Raises the **pasting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPastingCell

```
onPastingCell(e: CellRangeEventArgs): boolean
```

Raises the **pastingCell** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onPrepareCellForEdit

```
onPrepareCellForEdit(e: CellRangeEventArgs): void
```

Raises the **prepareCellForEdit** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onPrepareChangingColumn

```
onPrepareChangingColumn(): void
```

Raises the prepareChangingColumn event.

Inherited From
FlexSheet
Returns
void

onPrepareChangingRow

```
onPrepareChangingRow(): void
```

Raises the prepareChangingRow event.

Inherited From
FlexSheet
Returns
void

onResizedColumn

```
onResizedColumn(e: CellRangeEventArgs): void
```

Raises the **resizedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizedRow

```
onResizedRow(e: CellRangeEventArgs): void
```

Raises the **resizedRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onResizingColumn

```
onResizingColumn(e: CellRangeEventArgs): boolean
```

Raises the **resizingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onResizingRow

```
onResizingRow(e: CellRangeEventArgs): boolean
```

Raises the **resizingRow** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowAdded

```
onRowAdded(e: CellRangeEventArgs): boolean
```

Raises the **rowAdded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onRowChanged

```
onRowChanged(e: RowColumnChangedEventArgs): void
```

Raises the rowChanged event.

Parameters

- **e**: **RowColumnChangedEventArgs**

Inherited From
FlexSheet
Returns
void

onRowEditEnded

```
onRowEditEnded(e: CellRangeEventArgs): void
```

Raises the **rowEditEnded** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditEnding

```
onRowEditEnding(e: CellRangeEventArgs): void
```

Raises the **rowEditEnding** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarted

```
onRowEditStarted(e: CellRangeEventArgs): void
```

Raises the **rowEditStarted** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onRowEditStarting

```
onRowEditStarting(e: CellRangeEventArgs): void
```

Raises the **rowEditStarting** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onScrollPositionChanged

```
onScrollPositionChanged(e?: EventArgs): void
```

Raises the **scrollPositionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onSelectedSheetChanged

```
onSelectedSheetChanged(e: PropertyChangedEventArgs): void
```

Raises the currentSheetChanged event.

Parameters

- **e**: **PropertyChangedEventArgs**
  PropertyChangedEventArgs that contains the event data.

Inherited From
FlexSheet
Returns
void

onSelectionChanged

```
onSelectionChanged(e: CellRangeEventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSelectionChanging

```
onSelectionChanging(e: CellRangeEventArgs): boolean
```

Raises the **selectionChanging** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onSheetCleared

```
onSheetCleared(e?: EventArgs): void
```

Raises the sheetCleared event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexSheet
Returns
void

onSortedColumn

```
onSortedColumn(e: CellRangeEventArgs): void
```

Raises the **sortedColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
void

onSortingColumn

```
onSortingColumn(e: CellRangeEventArgs): boolean
```

Raises the **sortingColumn** event.

Parameters

- **e**: **CellRangeEventArgs**
  CellRangeEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUnknownFunction

```
onUnknownFunction(e: UnknownFunctionEventArgs): void
```

Raises the unknownFunction event.

Parameters

- **e**: **UnknownFunctionEventArgs**

Inherited From
FlexSheet
Returns
void

onUpdatedLayout

```
onUpdatedLayout(e?: EventArgs): void
```

Raises the **updatedLayout** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatedView

```
onUpdatedView(e?: EventArgs): void
```

Raises the **updatedView** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexGrid
Returns
void

onUpdatingLayout

```
onUpdatingLayout(e: CancelEventArgs): boolean
```

Raises the **updatingLayout** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

onUpdatingView

```
onUpdatingView(e: CancelEventArgs): boolean
```

Raises the **updatingView** event.

Parameters

- **e**: **CancelEventArgs**
  CancelEventArgs that contains the event data.

Inherited From
FlexGrid
Returns
boolean

redo

```
redo(): void
```

Redo the last user action.

Inherited From
FlexSheet
Returns
void

refresh

```
refresh(fullUpdate?: boolean): void
```

Overridden to refresh the sheet and the TabHolder.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
FlexSheet
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **invalidateAll**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

refreshCells

```
refreshCells(fullUpdate: boolean, recycle?: boolean, state?: boolean): void
```

Refreshes the grid display.

Parameters

- **fullUpdate**: **boolean**
  Whether to update the grid layout and content, or just the content.
- **recycle**: **boolean** OPTIONAL
  Whether to recycle existing elements.
- **state**: **boolean** OPTIONAL
  Whether to keep existing elements and update their state.

Inherited From
`FlexGrid`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **`Control`**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
`Control`
Returns
number

save

```
save(fileName?: string): Workbook
```

Saves **FlexSheet** to xlsx file. This method works with JSZip 2.5.

For example:

```
// This sample exports FlexSheet content to an xlsx file.
// click.


// HTML
<button
    onclick="saveXlsx('FlexSheet.xlsx')">
    Save
</button>


// JavaScript
function saveXlsx(fileName) {

    // Save the flexGrid to xlsx file.
    flexsheet.save(fileName);
}
```

Parameters

- **fileName**: **string** OPTIONAL
  Name of the file that is generated.

Inherited From
`FlexSheet`
Returns
`Workbook`

saveAsync

```
saveAsync(fileName?: string, onSaved?: (base64?: string), onError?: (reason?: any)): void
```

Saves the **FlexSheet** to xlsx file asynchronously. This method works with JSZip 3.0.

Parameters

- **fileName**: **string** OPTIONAL
  Name of the file that is generated.

- **onSaved**: **(base64?: string)** OPTIONAL
  This callback provides an approach to get the base-64 string that represents the content of the saved FlexSheet. Since this method is an asynchronous method, user is not able to get the base-64 string immediately. User has to get the base-64 string through this callback. This has a single parameter, the base64 string of the saved flexsheet. It is passed to user.

- **onError**: **(reason?: any)** OPTIONAL
  This callback catches error information when saving. This has a single parameter, the failure reason. The return value is passed to user if he wants to catch the save failure reason.

  For example:

```
flexsheet.saveAsync('', function (base64) {

     // user can access the base64 string in this callback.
     document.getElementById('export').href = 'data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;' + 'base64,' + base64;
}, function (reason) {

     // User can catch the failure reason in this callback.
     console.log('The reason of save failure is ' + reason);
});
```

Inherited From
`FlexSheet`
Returns
void

scrollIntoView

```
scrollIntoView(r: number, c: number): boolean
```

Scrolls the grid to bring a specific cell into view.

Negative row and column indices are ignored, so if you call

```
grid.scrollIntoView(200, -1);
```

The grid will scroll vertically to show row 200, and will not scroll horizontally.

Parameters

- **r**: **number**
  Index of the row to scroll into view.

- **c**: **number**
  Index of the column to scroll into view.

Inherited From
`FlexGrid`
Returns
boolean

select

```
select(rng: any, show?: any): void
```

Selects a cell range and optionally scrolls it into view.

FlexSheet overrides this method to adjust the selection cell range for the merged cells in the **FlexSheet**.

Parameters

- **rng**: **any**
  The cell range to select.

- **show**: **any** OPTIONAL
  Indicates whether to scroll the new selection into view.

Inherited From
FlexSheet
Returns
void

selectNextFunction

```
selectNextFunction(): void
```

Select next function in the function list.

Inherited From
FlexSheet
Returns
void

selectPreviousFunction

```
selectPreviousFunction(): void
```

Select previous function in the function list.

Inherited From
`FlexSheet`
Returns
void

setCellData

```
setCellData(r: number, c: any, value: any, coerce?: boolean): boolean
```

Overrides the setCellData function of the base class.

Parameters

- **r**: **number**
  Index of the row that contains the cell.
- **c**: **any**
  Index, name, or binding of the column that contains the cell.
- **value**: **any**
  Value to store in the cell.
- **coerce**: **boolean** OPTIONAL
  Whether to change the value automatically to match the column's data type.

Inherited From
`FlexSheet`
Returns
boolean

setClipString

```
setClipString(text: string, rng?: CellRange): void
```

Parses a string into rows and columns and applies the content to a given range.

Override the **setClipString** method of **FlexGrid**.

Parameters

- **text**: **string**
  Tab and newline delimited text to parse into the grid.

- **rng**: **CellRange** OPTIONAL
  CellRange to copy. If omitted, the current selection is used.

Inherited From
FlexSheet
Returns
void

showColumnFilter

```
showColumnFilter(): void
```

Show the filter editor.

Inherited From
FlexSheet
Returns
void

showFunctionList

```
showFunctionList(target: HTMLElement): void
```

Open the function list.

Parameters

- **target**: **HTMLElement**
  The DOM element that toggle the function list.

Inherited From
FlexSheet
Returns
void

startEditing

```
startEditing(fullEdit?: boolean, r?: number, c?: number, focus?: boolean): boolean
```

Starts editing a given cell.

Editing in the **FlexGrid** is similar to editing in Excel: Pressing F2 or double-clicking a cell puts the grid in **full-edit** mode. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or until he moves the selection with the mouse. In full-edit mode, pressing the cursor keys does not cause the grid to exit edit mode.

Typing text directly into a cell puts the grid in **quick-edit mode**. In this mode, the cell editor remains active until the user presses Enter, Tab, or Escape, or any arrow keys.

Full-edit mode is normally used to make changes to existing values. Quick-edit mode is normally used for entering new data quickly.

While editing, the user can toggle between full and quick modes by pressing the F2 key.

Parameters

- **fullEdit**: **boolean** OPTIONAL
  Whether to stay in edit mode when the user presses the cursor keys. Defaults to true.
- **r**: **number** OPTIONAL
  Index of the row to be edited. Defaults to the currently selected row.
- **c**: **number** OPTIONAL
  Index of the column to be edited. Defaults to the currently selected column.
- **focus**: **boolean** OPTIONAL
  Whether to give the editor the focus when editing starts. Defaults to true.

Inherited From
FlexGrid
Returns
boolean

toggleDropDownList

```
toggleDropDownList(): void
```

Toggles the drop-down list-box associated with the currently selected cell.

This method can be used to show the drop-down list automatically when the cell enters edit mode, or when the user presses certain keys.

For example, this code causes the grid to show the drop-down list whenever the grid enters edit mode:

```
// show the drop-down list when the grid enters edit mode
theGrid.beginningEdit = function () {
  theGrid.toggleDropDownList();
}
```

This code causes the grid to show the drop-down list when the grid enters edit mode after the user presses the space bar:

```
// show the drop-down list when the user presses the space bar
theGrid.hostElement.addEventListener('keydown', function (e) {
  if (e.keyCode == 32) {
    e.preventDefault();
    theGrid.toggleDropDownList();
  }
}, true);
```

Inherited From
FlexGrid
Returns
void

undo

```
undo(): void
```

Undo the last user action.

Inherited From
FlexSheet
Returns
void

Events

autoSizedColumn

Occurs after the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizedRow

Occurs after the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingColumn

Occurs before the user auto-sizes a column by double-clicking the right edge of a column header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

autoSizingRow

Occurs before the user auto-sizes a row by double-clicking the bottom edge of a row header cell.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

beginningEdit

Occurs before a cell enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnded

Occurs when a cell edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

cellEditEnding

Occurs when a cell edit is ending.

You can use this event to perform validation and prevent invalid edits. For example, the code below prevents users from entering values that do not contain the letter 'a'. The code demonstrates how you can obtain the old and new values before the edits are applied.

```
function cellEditEnding (sender, e) {

  // get old and new values
  var flex = sender,
      oldVal = flex.getCellData(e.row, e.col),
      newVal = flex.activeEditor.value;

  // cancel edits if newVal doesn't contain 'a'
  e.cancel = newVal.indexOf('a') < 0;
}
```

Setting the **`cancel`** parameter to true causes the grid to discard the edited value and keep the cell's original value.

If you also set the **`stayInEditMode`** parameter to true, the grid will remain in edit mode so the user can correct invalid entries before committing the edits.

Inherited From
`FlexGrid`
Arguments
`CellEditEndingEventArgs`

columnChanged

Occurs after the **FlexSheet** insert\delete columns.

Inherited From
FlexSheet
Arguments
RowColumnChangedEventArgs

copied

Occurs after the user has copied the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

copying

Occurs when the user is copying the selection content to the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletedRow

Occurs after the user has deleted a row by pressing the Delete key (see the **allowDelete** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

deletingRow

Occurs when the user is deleting a selected row by pressing the Delete key (see the **`allowDelete`** property).

The event handler may cancel the row deletion.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggedColumn

Occurs when the user finishes dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggedRow

Occurs when the user finishes dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumn

Occurs when the user starts dragging a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingColumnOver

Occurs as the user drags a column to a new position.

The handler may cancel the event to prevent users from dropping columns at certain positions. For example:

```
// remember column being dragged
flex.draggingColumn.addHandler(function (s, e) {
    theColumn = s.columns[e.col].binding;
});

// prevent 'sales' column from being dragged to index 0
s.draggingColumnOver.addHandler(function (s, e) {
    if (theColumn == 'sales' && e.col == 0) {
        e.cancel = true;
    }
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRow

Occurs when the user starts dragging a row.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

draggingRowColumn

Occurs when dragging the rows or the columns of the **FlexSheet**.

Inherited From
`FlexSheet`
Arguments
`DraggingRowColumnEventArgs`

draggingRowOver

Occurs as the user drags a row to a new position.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

droppingRowColumn

Occurs when dropping the rows or the columns of the **FlexSheet**.

Inherited From
`FlexSheet`
Arguments
`EventArgs`

formatItem

Occurs when an element representing a cell has been created.

This event can be used to format cells for display. It is similar in purpose to the **`itemFormatter`** property, but has the advantage of allowing multiple independent handlers.

For example, this code removes the 'wj-wrap' class from cells in group rows:

```
flex.formatItem.addHandler(function (s, e) {
  if (flex.rows[e.row] instanceof wijmo.grid.GroupRow) {
    wijmo.removeClass(e.cell, 'wj-wrap');
  }
});
```

Inherited From
`FlexGrid`
Arguments
`FormatItemEventArgs`

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

groupCollapsedChanged

Occurs after a group has been expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

groupCollapsedChanging

Occurs when a group is about to be expanded or collapsed.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

itemsSourceChanged

Occurs after the grid has been bound to a new items source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loaded

Occurs after the **`FlexSheet`** loads the **`Workbook`** instance

Inherited From
`FlexSheet`
Arguments
`EventArgs`

loadedRows

Occurs after the grid rows have been bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

loadingRows

Occurs before the grid rows are bound to items in the data source.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

pasted

Occurs after the user has pasted content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastedCell

Occurs after the user has pasted content from the clipboard into a cell (see the **autoClipboard** property).

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pasting

Occurs when the user is pasting content from the clipboard by pressing one of the clipboard shortcut keys (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

pastingCell

Occurs when the user is pasting content from the clipboard into a cell (see the **autoClipboard** property).

The event handler may cancel the copy operation.

Inherited From
FlexGrid
Arguments
CellRangeEventArgs

prepareCellForEdit

Occurs when an editor cell is created and before it becomes active.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

prepareChangingColumn

Occurs before the **`FlexSheet`** insert\delete columns.

Inherited From
`FlexSheet`
Arguments
`EventArgs`

prepareChangingRow

Occurs before the **`FlexSheet`** insert\delete rows.

Inherited From
`FlexSheet`
Arguments
`EventArgs`

resizedColumn

Occurs when the user finishes resizing a column.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizedRow

Occurs when the user finishes resizing rows.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingColumn

Occurs as columns are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

resizingRow

Occurs as rows are resized.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowAdded

Occurs when the user creates a new item by editing the new row template (see the **`allowAddNew`** property).

The event handler may customize the content of the new item or cancel the new item creation.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowChanged

Occurs after the **`FlexSheet`** insert\delete rows.

Inherited From
`FlexSheet`
Arguments
`RowColumnChangedEventArgs`

rowEditEnded

Occurs when a row edit has been committed or canceled.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditEnding

Occurs when a row edit is ending, before the changes are committed or canceled.

This event can be used in conjunction with the **`rowEditStarted`** event to implement deep-binding edit undos. For example:

```
// save deep bound values when editing starts
var itemData = {};
s.rowEditStarted.addHandler(function (s, e) {
  var item = s.collectionView.currentEditItem;
  itemData = {};
  s.columns.forEach(function (col) {
    if (col.binding.indexOf('.') > -1) { // deep binding
      var binding = new wijmo.Binding(col.binding);
      itemData[col.binding] = binding.getValue(item);
    }
  })
});

// restore deep bound values when edits are canceled
s.rowEditEnded.addHandler(function (s, e) {
  if (e.cancel) { // edits were canceled by the user
    var item = s.collectionView.currentEditItem;
    for (var k in itemData) {
      var binding = new wijmo.Binding(k);
      binding.setValue(item, itemData[k]);
    }
  }
  itemData = {};
});
```

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarted

Occurs after a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

rowEditStarting

Occurs before a row enters edit mode.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

scrollPositionChanged

Occurs after the control has scrolled.

Inherited From
`FlexGrid`
Arguments
`EventArgs`

selectedSheetChanged

Occurs when current sheet index changed.

Inherited From
`FlexSheet`
Arguments
`PropertyChangedEventArgs`

selectionChanged

Occurs after selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

selectionChanging

Occurs before selection changes.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sheetCleared

Occurs when the `FlexSheet` is cleared.

Inherited From
`FlexSheet`
Arguments
`EventArgs`

sortedColumn

Occurs after the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

sortingColumn

Occurs before the user applies a sort by clicking on a column header.

Inherited From
`FlexGrid`
Arguments
`CellRangeEventArgs`

unknownFunction

Occurs when the **FlexSheet** meets the unknown formula.

Inherited From
FlexSheet
Arguments
UnknownFunctionEventArgs

updatedLayout

Occurs after the grid has updated its internal layout.

Inherited From
FlexGrid
Arguments
EventArgs

updatedView

Occurs when the grid finishes creating/updating the elements that make up the current view.

The grid updates the view in response to several actions, including:

- refreshing the grid or its data source,
- adding, removing, or changing rows or columns,
- resizing or scrolling the grid,
- changing the selection.

Inherited From
FlexGrid
Arguments
EventArgs

updatingLayout

Occurs before the grid updates its internal layout.

Inherited From
FlexGrid
Arguments
CancelEventArgs

updatingView

Occurs when the grid starts creating/updating the elements that make up the current view.

Inherited From
`FlexGrid`
Arguments
`CancelEventArgs`

WjSheet Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.grid.sheet
Base Class
Sheet

Angular 2 component for the **Sheet** control.

The **wj-sheet** component must be contained in a **WjFlexSheet** component.

Use the **wj-sheet** component to add **Sheet** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjSheet** component is derived from the **Sheet** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- columnCount
- grid
- initialized
- isInitialized
- itemsSource
- name
- nameChangedNg
- rowCount
- selectionRanges
- tableNames
- visible
- wjProperty

Methods

- created
- findTable
- getCellStyle
- onNameChanged
- onVisibleChanged

Events

- nameChanged
- visibleChanged

Constructor

constructor

```
constructor(owner?: FlexSheet, grid?: FlexGrid, sheetName?: string, rows?: number, cols?: number): Sheet
```

Initializes a new instance of the **Sheet** class.

Parameters

- **owner**: **FlexSheet** OPTIONAL
  The owner @see: FlexSheet control.
- **grid**: **FlexGrid** OPTIONAL
  The associated **FlexGrid** control used to store the sheet data. If not specified then the new **FlexGrid** control will be created.
- **sheetName**: **string** OPTIONAL
  The name of the sheet within the **FlexSheet** control.
- **rows**: **number** OPTIONAL
  The row count for the sheet.
- **cols**: **number** OPTIONAL
  The column count for the sheet.

Inherited From
Sheet
Returns
Sheet

Properties

columnCount

Gets or sets the number of columns in the sheet.

Inherited From
Sheet
Type
number

grid

Gets the associated **FlexGrid** control used to store the sheet data.

Inherited From
Sheet
Type
FlexGrid

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or **ICollectionView** for the **FlexGrid** instance of the sheet.

Inherited From
Sheet
Type
any

name

Gets or sets the name of the sheet.

Inherited From
Sheet
Type
string

nameChangedNg

Angular (EventEmitter) version of the Wijmo **nameChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **nameChanged** Wijmo event name.

Type
EventEmitter

rowCount

Gets or sets the number of rows in the sheet.

Inherited From
Sheet
Type
number

selectionRanges

Gets the selection array.

Inherited From
Sheet
Type
ObservableArray

tableNames

Gets the names of tables render in this sheet.

Inherited From
Sheet
Type
string[]

visible

Gets or sets the sheet visibility.

Inherited From
Sheet
Type
boolean

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

findTable

```
findTable(rowIndex: number, columnIndex: number): Table
```

Finds the table via the cell location.

Parameters

- **rowIndex**: **number**
  the row index of the specified cell.

- **columnIndex**: **number**
  the column index of the specified cell.

Inherited From
Sheet
Returns
Table

getCellStyle

```
getCellStyle(rowIndex: number, columnIndex: number): ICellStyle
```

Gets the style of specified cell.

Parameters

- **rowIndex**: **number**
  the row index of the specified cell.

- **columnIndex**: **number**
  the column index of the specified cell.

Inherited From
Sheet
Returns
ICellStyle

onNameChanged

```
onNameChanged(e: PropertyChangedEventArgs): void
```

Raises the **nameChanged** event.

Parameters

- **e**: **PropertyChangedEventArgs**

Inherited From
Sheet
Returns
void

onVisibleChanged

```
onVisibleChanged(e: EventArgs): void
```

Raises the **visibleChanged** event.

Parameters

- **e**: **EventArgs**

Inherited From
Sheet
Returns
void

Events

nameChanged

Occurs after the sheet name has changed.

Inherited From
Sheet
Arguments
PropertyChangedEventArgs

visibleChanged

Occurs after the visible of sheet has changed.

Inherited From
Sheet
Arguments
EventArgs

wijmo/wijmo.angular2.chart Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart

Contains Angular 2 components for the **wijmo.chart** module.

wijmo.angular2.chart is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFlexChart
- WjFlexChartAxis
- WjFlexChartDataLabel
- WjFlexChartDataPoint
- WjFlexChartLegend
- WjFlexChartLineMarker
- WjFlexChartPlotArea
- WjFlexChartSeries
- WjFlexPie
- WjFlexPieDataLabel

WjFlexChart Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
FlexChart

Angular 2 component for the **FlexChart** control.

Use the **wj-flex-chart** component to add **FlexChart** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChart** component is derived from the **FlexChart** control and inherits all its properties, events and methods.

The **wj-flex-chart** component may contain the following child components: **WjFlexChartTrendLine** , **WjFlexChartMovingAverage** , **WjFlexChartYFunctionSeries** , **WjFlexChartParametricFunctionSeries** , **WjFlexChartWaterfall** , **WjFlexChartBoxWhisker** , **WjFlexChartErrorBar** , **WjFlexChartAnimation** , **WjFlexChartAnnotationLayer** , **WjFlexChartRangeSelector** , **WjFlexChartGestures** , **WjFlexChartAxis** , **WjFlexChartLegend** , **WjFlexChartDataLabel** , **WjFlexChartSeries** , **WjFlexChartLineMarker** and **WjFlexChartPlotArea**.

Constructor

- constructor

Properties

- asyncBindings
- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- interpolateNulls
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- lostFocusNg
- options
- palette
- plotAreas
- plotMargin
- renderedNg
- renderingNg
- rightToLeft
- rotated
- selection
- selectionChangedNg
- selectionMode
- series
- seriesVisibilityChangedNg
- stacking
- symbolSize
- tooltip
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered

- rendering
- selectionChanged
- seriesVisibilityChanged

Constructor

constructor

```
constructor(element: any, options?): FlexChart
```

Initializes a new instance of the **FlexChart** class.

Parameters

- **element**: **any**
  The DOM element that will host the control, or a selector for the host element (e.g. '#theCtrl').
- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Inherited From
FlexChart
Returns
FlexChart

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axes

Gets the collection of **Axis** objects.

Inherited From
FlexChartCore
Type
ObservableArray

axisX

Gets or sets the main X axis.

Inherited From
`FlexChartCore`
Type
`Axis`

axisY

Gets or sets the main Y axis.

Inherited From
`FlexChartCore`
Type
`Axis`

binding

Gets or sets the name of the property that contains the Y values.

Inherited From
`FlexChartCore`
Type
string

bindingX

Gets or sets the name of the property that contains the X data values.

Inherited From
`FlexChartCore`
Type
string

chartType

Gets or sets the type of chart to create.

Inherited From
`FlexChart`
Type
`ChartType`

collectionView

Gets the `ICollectionView` object that contains the chart data.

Inherited From
`FlexChartBase`
Type
`ICollectionView`

dataLabel

Gets or sets the point data label.

Inherited From
`FlexChartCore`
Type
`DataLabel`

footer

Gets or sets the text displayed in the chart footer.

Inherited From
`FlexChartBase`
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
`FlexChartBase`
Type
any

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

interpolateNulls

Gets or sets a value that determines whether to interpolate null values in the data.

If true, the chart interpolates the value of any missing data based on neighboring points. If false, it leaves a break in lines and areas at the points with null values.

Inherited From
`FlexChartCore`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the data used to create the chart.

Inherited From
FlexChartBase
Type
any

legend

Gets or sets the chart legend.

Inherited From
FlexChartBase
Type
Legend

legendToggle

Gets or sets a value indicating whether clicking legend items toggles the series visibility in the chart.

Inherited From
FlexChartCore
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

options

Gets or sets various chart options.

The following options are supported:

bubble.maxSize: Specifies the maximum size of symbols in the Bubble chart. The default value is 30 pixels.

bubble.minSize: Specifies the minimum size of symbols in the Bubble chart. The default value is 5 pixels.

```
chart.options = {
  bubble: { minSize: 5, maxSize: 30 }
}
```

funnel.neckWidth: Specifies the neck width as a percentage for the Funnel chart. The default value is 0.2.

funnel.neckHeight: Specifies the neck height as a percentage for the Funnel chart. The default value is 0.

funnel.type: Specifies the type of Funnel chart. It should be 'rectangle' or 'default'. neckWidth and neckHeight don't work if type is set to rectangle.

```
chart.options = {
  funnel: { neckWidth: 0.3, neckHeight: 0.3, type: 'rectangle' }
}
```

groupWidth: Specifies the group width for the Column charts, or the group height for the Bar charts. The group width can be specified in pixels or as percentage of the available space. The default value is '70%'.

```
chart.options = {
  groupWidth : 50; // 50 pixels
}
chart.options = {
  groupWidth : '100%'; // 100% pixels
}
```

Inherited From
`FlexChart`
Type
any

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **Palettes** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
FlexChartBase
Type
string[]

plotAreas

Gets the collection of **PlotArea** objects.

Inherited From
FlexChartCore
Type
PlotAreaCollection

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

rotated

Gets or sets a value indicating whether to flip the axes so that X becomes vertical and Y becomes horizontal.

Inherited From
`FlexChart`
Type
boolean

selection

Gets or sets the selected chart series.

Inherited From
`FlexChartCore`
Type
`SeriesBase`

selectionChangedNg

Angular (EventEmitter) version of the Wijmo **selectionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanged** Wijmo event name.

Type
EventEmitter

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

series

Gets the collection of **`Series`** objects.

Inherited From
`FlexChartCore`
Type
`ObservableArray`

seriesVisibilityChangedNg

Angular (EventEmitter) version of the Wijmo **seriesVisibilityChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **seriesVisibilityChanged** Wijmo event name.

Type
EventEmitter

stacking

Gets or sets a value that determines whether and how the series objects are stacked.

Inherited From
`FlexChart`
Type
`Stacking`

symbolSize

Gets or sets the size of the symbols used for all Series objects in this **FlexChart**.

This property may be overridden by the symbolSize property on each **Series** object.

Inherited From
FlexChartCore
Type
number

tooltip

Gets the chart **Tooltip** object.

The tooltip content is generated using a template that may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

To modify the template, assign a new value to the tooltip's content property. For example:

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

You can disable chart tooltips by setting the template to an empty string.

You can also use the **tooltip** property to customize tooltip parameters such as **showDelay** and **hideDelay**:

```
chart.tooltip.showDelay = 1000;
```

See **ChartTooltip** properties for more details and options.

Inherited From
FlexChartCore
Type
ChartTooltip

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Converts a **`Point`** from data coordinates to control coordinates.

Parameters

- **pt**: **any**
  `Point` in data coordinates, or X coordinate of a point in data coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

Raises the **seriesVisibilityChanged** event.

Parameters

- **e**: **SeriesEventArgs**
  The **SeriesEventArgs** object that contains the event data.

Inherited From
FlexChartCore
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
`FlexChartBase`
Returns
`Point`

pointToData

```
pointToData(pt: any, y?: number): Point
```

Converts a **`Point`** from control coordinates to chart data coordinates.

Parameters

- **pt**: **any**
  The point to convert, in control coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

seriesVisibilityChanged

Occurs when the series visibility changes, for example when the legendToggle property is set to true and the user clicks the legend.

Inherited From
`FlexChartCore`
Arguments
`SeriesEventArgs`

WjFlexChartAxis Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
Axis

Angular 2 component for the **Axis** control.

The **wj-flex-chart-axis** component must be contained in one of the following components: **WjFlexChart** , **WjFlexChartSeries** , **WjFinancialChart** or **WjFinancialChartSeries**.

Use the **wj-flex-chart-axis** component to add **Axis** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAxis** component is derived from the **Axis** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- actualMax
- actualMin
- axisLine
- axisType
- binding
- format
- hostElement
- initialized
- isInitialized
- itemFormatter
- itemsSource
- labelAlign
- labelAngle
- labelPadding
- labels
- logBase
- majorGrid
- majorTickMarks
- majorUnit
- max
- min
- minorGrid
- minorTickMarks
- minorUnit
- name
- origin
- overlappingLabels
- plotArea
- position
- rangeChangedNg
- reversed
- title
- wjProperty

Methods

- convert
- convertBack
- created
- onRangeChanged

Events

- rangeChanged

Constructor

constructor

```
constructor(position?: Position): Axis
```

Initializes a new instance of the **Axis** class.

Parameters

- **position**: **Position** OPTIONAL
  The position of the axis on the chart.

Inherited From
Axis
Returns
Axis

Properties

actualMax

Gets the actual axis maximum.

It returns a number or a Date object (for time-based data).

Inherited From
Axis
Type
any

actualMin

Gets the actual axis minimum.

It returns a number or a Date object (for time-based data).

Inherited From
Axis
Type
any

axisLine

Gets or sets a value indicating whether the axis line is visible.

Inherited From
`Axis`
Type
boolean

axisType

Gets the axis type.

Inherited From
`Axis`
Type
`AxisType`

binding

Gets or sets the comma-separated property names for the **`itemsSource`** property to use in axis labels.

The first name specifies the value on the axis, the second represents the corresponding axis label. The default value is 'value,text'.

Inherited From
`Axis`
Type
string

format

Gets or sets the format string used for the axis labels (see **`Globalize`**).

Inherited From
`Axis`
Type
string

hostElement

Gets the axis host element.

Inherited From
`Axis`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemFormatter

Gets or sets the itemFormatter function for the axis labels.

If specified, the function takes two parameters:

- **render engine**: The **IRenderEngine** object to be used in formatting the labels.
- **current label**: An object with the following properties:
  - **value**: The value of the axis label to format.
  - **text**: The text to use in the label.
  - **pos**: The position in control coordinates at which the label is to be rendered.
  - **cls**: The CSS class to be applied to the label.

The function returns the label parameters of labels for which properties are modified.

For example:

```
chart.axisY.itemFormatter = function(engine, label) {
    if (label.val > 5){
        engine.textFill = 'red'; // red text
        label.cls = null; // no default CSS
     }
    return label;
}
```

Inherited From
`Axis`
Type
Function

itemsSource

Gets or sets the items source for the axis labels.

Names of the properties are specified by the **`binding`** property.

For example:

```
// default value for Axis.binding is 'value,text'
 chart.axisX.itemsSource = [ { value:1, text:'one' }, { value:2, text:'two' } ];
```

Inherited From
`Axis`
Type
any

labelAlign

Gets or sets the label alignment.

By default the labels are centered. The supported values are 'left' and 'right for x-axis and 'top' and 'bottom' for y-axis.

Inherited From
`Axis`
Type
string

labelAngle

Gets or sets the rotation angle of the axis labels.

The angle is measured in degrees with valid values ranging from -90 to 90.

Inherited From
`Axis`
Type
number

labelPadding

Gets or sets the label padding.

Inherited From
`Axis`
Type
number

labels

Gets or sets a value indicating whether the axis labels are visible.

Inherited From
`Axis`
Type
boolean

logBase

Gets or sets the logarithmic base of the axis.

If the base is not specified the axis uses a linear scale.

Use the **`logBase`** property to spread data that is clustered around the origin. This is common in several financial and economic data sets.

Inherited From
`Axis`
Type
number

majorGrid

Gets or sets a value indicating whether the axis includes grid lines.

Inherited From
`Axis`
Type
boolean

majorTickMarks

Gets or sets the location of the axis tick marks.

Inherited From
`Axis`
Type
`TickMark`

majorUnit

Gets or sets the number of units between axis labels.

If the axis contains date values, then the units are expressed in days.

Inherited From
`Axis`
Type
number

max

Gets or sets the maximum value shown on the axis.

If not set, the maximum is calculated automatically. The value can be a number or a Date object (for time-based data).

Inherited From
`Axis`
Type
any

min

Gets or sets the minimum value shown on the axis.

If not set, the minimum is calculated automatically. The value can be a number or a Date object (for time-based data).

Inherited From
`Axis`
Type
any

minorGrid

Gets or sets a value indicating whether the axis includes minor grid lines.

Inherited From
`Axis`
Type
boolean

minorTickMarks

Gets or sets the location of the minor axis tick marks.

Inherited From
`Axis`
Type
`TickMark`

minorUnit

Gets or sets the number of units between minor axis ticks.

If the axis contains date values, then the units are expressed in days.

Inherited From
`Axis`
Type
number

name

Gets or sets the axis name.

Inherited From
`Axis`
Type
string

origin

Gets or sets the value at which an axis crosses the perpendicular axis.

Inherited From
`Axis`
Type
number

overlappingLabels

Gets or sets a value indicating how to handle the overlapping axis labels.

Inherited From
`Axis`
Type
`OverlappingLabels`

plotArea

Gets or sets the plot area for the axis.

Inherited From
`Axis`
Type
`PlotArea`

position

Gets or sets the position of the axis with respect to the plot area.

Inherited From
`Axis`
Type
`Position`

- **rangeChangedNg**

Angular (EventEmitter) version of the Wijmo **rangeChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rangeChanged** Wijmo event name.

Type
EventEmitter

- **reversed**

Gets or sets a value indicating whether the axis is reversed (top to bottom or right to left).

Inherited From
`Axis`
Type
boolean

- **title**

Gets or sets the title text shown next to the axis.

Inherited From
`Axis`
Type
string

- **wjProperty**

Gets or sets a name of a property that this component is assigned to. Default value is 'axes'.

Type
string

Methods

convert

```
convert(val: number, maxValue?: number, minValue?: number): number
```

Converts the specified value from data to pixel coordinates.

Parameters

- **val**: **number**
  The data value to convert.
- **maxValue**: **number** OPTIONAL
  The max value of the data, it's optional.
- **minValue**: **number** OPTIONAL
  The min value of the data, it's optional.

Inherited From
Axis
Returns
number

convertBack

```
convertBack(val: number): number
```

Converts the specified value from pixel to data coordinates.

Parameters

- **val**: **number**
  The pixel coordinates to convert back.

Inherited From
Axis
Returns
number

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

Raises the **rangeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Axis
Returns
void

Events

rangeChanged

Occurs when the axis range changes.

Inherited From
Axis
Arguments
EventArgs

WjFlexChartDataLabel Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
DataLabel

Angular 2 component for the **DataLabel** control.

The **wj-flex-chart-data-label** component must be contained in a **WjFlexChart** component.

Use the **wj-flex-chart-data-label** component to add **DataLabel** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartDataLabel** component is derived from the **DataLabel** control and inherits all its properties, events and methods.

Properties

- border
- connectingLine
- content
- initialized
- isInitialized
- offset
- position
- renderingNg
- wjProperty

Methods

- created
- onRendering

Events

- rendering

Properties

border

Gets or sets a value indicating whether the data labels have borders.

Inherited From
`DataLabelBase`
Type
boolean

connectingLine

Gets or sets a value indicating whether to draw lines that connect labels to the data points.

Inherited From
`DataLabelBase`
Type
boolean

content

Gets or sets the content of data labels.

The content can be specified as a string or as a function that takes **HitTestInfo** object as a parameter.

When the label content is a string, it can contain any of the following parameters:

- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point.
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point.
- **propertyName**: any property of data object.

The parameter must be enclosed in curly brackets, for example 'x={x}, y={y}'.

In the following example, we show the y value of the data point in the labels.

```
// Create a chart and show y data in labels positioned above the data point.
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

The next example shows how to set data label content using a function.

```
// Set the data label content
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

Inherited From
DataLabelBase
Type
any

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

offset

Gets or sets the offset from label to the data point.

Inherited From
`DataLabelBase`
Type
number

position

Gets or sets the position of the data labels.

Inherited From
`DataLabel`
Type
`LabelPosition`

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'dataLabel'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

Raises the **`rendering`** event.

Parameters

- **e**: **DataLabelRenderEventArgs**
  The **`DataLabelRenderEventArgs`** object used to render the label.

Inherited From
`DataLabelBase`
Returns
void

Events

rendering

Occurs before the data label is rendered.

Inherited From
`DataLabelBase`
Arguments
`DataLabelRenderEventArgs`

WjFlexChartDataPoint Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
DataPoint

Angular 2 component for the **DataPoint** control.

The **wj-flex-chart-data-point** component must be contained in one of the following components: **WjFlexChartAnnotationText** , **WjFlexChartAnnotationEllipse** , **WjFlexChartAnnotationRectangle** , **WjFlexChartAnnotationLine** , **WjFlexChartAnnotationPolygon** , **WjFlexChartAnnotationCircle** , **WjFlexChartAnnotationSquare** or **WjFlexChartAnnotationImage**.

Use the **wj-flex-chart-data-point** component to add **DataPoint** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartDataPoint** component is derived from the **DataPoint** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- initialized
- isInitialized
- wjProperty
- x
- y

Methods

- created

Constructor

constructor

```
constructor(x?: any, y?: any): DataPoint
```

Initializes a new instance of the **DataPoint** class.

Parameters

- **x**: **any** OPTIONAL
  X coordinate of the new DataPoint.

- **y**: **any** OPTIONAL
  Y coordinate of the new DataPoint.

Inherited From
DataPoint
Returns
DataPoint

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

x

Gets or sets X coordinate value of this **DataPoint**.

Inherited From
DataPoint
Type
any

y

Gets or sets Y coordinate value of this **DataPoint**.

Inherited From
DataPoint
Type
any

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjFlexChartLegend Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
Legend

Angular 2 component for the **Legend** control.

The **wj-flex-chart-legend** component must be contained in one of the following components: **WjFlexChart** , **WjFlexPie** , **WjFinancialChart** , **WjFlexRadar** or **WjSunburst**.

Use the **wj-flex-chart-legend** component to add **Legend** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartLegend** component is derived from the **Legend** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- initialized
- isInitialized
- position
- wjProperty

Methods

- created

Constructor

constructor

```
constructor(chart: FlexChartBase): Legend
```

Initializes a new instance of the **Legend** class.

Parameters

- **chart**: **FlexChartBase**
  `FlexChartBase` that owns this **Legend**.

Inherited From
Legend
Returns
Legend

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

position

Gets or sets a value that determines whether and where the legend appears in relation to the plot area.

Inherited From
`Legend`
Type
`Position`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'legend'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjFlexChartLineMarker Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
LineMarker

Angular 2 component for the **LineMarker** control.

The **wj-flex-line-marker** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-line-marker** component to add **LineMarker** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartLineMarker** component is derived from the **LineMarker** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- alignment
- chart
- content
- dragContent
- dragLines
- dragThreshold
- horizontalPosition
- initialized
- interaction
- isInitialized
- isVisible
- lines
- positionChangedNg
- seriesIndex
- verticalPosition
- wjProperty
- x
- y

Methods

- created
- onPositionChanged
- remove

Events

- positionChanged

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): LineMarker
```

Initializes a new instance of the **LineMarker** class.

Parameters

- **chart**: **FlexChartCore**
  The chart on which the LineMarker appears.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Inherited From
LineMarker
Returns
LineMarker

Properties

alignment

Gets or sets the alignment of the LineMarker content.

By default, the LineMarker shows to the right, at the bottom of the target point. Use '|' to combine alignment values.

```
// set the alignment to the left.
marker.alignment = wijmo.chart.LineMarkerAlignment.Left;

// set the alignment to the left top.
marker.alignment = wijmo.chart.LineMarkerAlignment.Left | wijmo.chart.LineMarkerAlignment.Top;
```

Inherited From
LineMarker
Type
LineMarkerAlignment

chart

Gets the **`FlexChart`** object that owns the LineMarker.

Inherited From
`LineMarker`
Type
`FlexChartCore`

content

Gets or sets the content function that allows you to customize the text content of the LineMarker.

Inherited From
`LineMarker`
Type
Function

dragContent

Gets or sets a value indicating whether the content of the marker is draggable when the interaction mode is "Drag."

Inherited From
`LineMarker`
Type
boolean

dragLines

Gets or sets a value indicating whether the lines are linked when the horizontal or vertical line is dragged when the interaction mode is "Drag."

Inherited From
`LineMarker`
Type
boolean

dragThreshold

Gets or sets the maximum distance from the horizontal or vertical line that the marker can be dragged.

Inherited From
`LineMarker`
Type
number

horizontalPosition

Gets or sets the horizontal position of the LineMarker relative to the plot area.

Its value range is (0, 1). If the value is null or undefined and **`interaction`** is set to wijmo.chart.LineMarkerInteraction.Move or wijmo.chart.LineMarkerInteraction.Drag, the horizontal position of the marker is calculated automatically based on the pointer's position.

Inherited From
`LineMarker`
Type
number

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

interaction

Gets or sets the interaction mode of the LineMarker.

Inherited From
`LineMarker`
Type
`LineMarkerInteraction`

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the LineMarker.

Inherited From
`LineMarker`
Type
boolean

lines

Gets or sets the visibility of the LineMarker lines.

Inherited From
`LineMarker`
Type
`LineMarkerLines`

positionChangedNg

Angular (EventEmitter) version of the Wijmo **positionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **positionChanged** Wijmo event name.

Type
EventEmitter

seriesIndex

Gets or sets the index of the series in the chart in which the LineMarker appears. This takes effect when the **`interaction`** property is set to wijmo.chart.LineMarkerInteraction.Move or wijmo.chart.LineMarkerInteraction.Drag.

Inherited From
`LineMarker`
Type
number

verticalPosition

Gets or sets the vertical position of the LineMarker relative to the plot area.

Its value range is (0, 1). If the value is null or undefined and **`interaction`** is set to wijmo.chart.LineMarkerInteraction.Move or wijmo.chart.LineMarkerInteraction.Drag, the vertical position of the LineMarker is calculated automatically based on the pointer's position.

Inherited From
`LineMarker`
Type
number

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

x

Gets the current x-value as chart data coordinates.

Inherited From
`LineMarker`
Type
number

y

Gets the current y-value as chart data coordinates.

Inherited From
`LineMarker`
Type
number

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

onPositionChanged

```
onPositionChanged(point: Point): void
```

Raises the **`positionChanged`** event.

Parameters

- **point**: **Point**
  The target point at which to show the LineMarker.

Inherited From
`LineMarker`
Returns
void

remove

```
remove(): void
```

Removes the LineMarker from the chart.

Inherited From
`LineMarker`
Returns
void

Events

positionChanged

Occurs after the **`LineMarker`**'s position changes.

Inherited From
`LineMarker`
Arguments
`Point`

WjFlexChartPlotArea Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
PlotArea

Angular 2 component for the **PlotArea** control.

The **wj-flex-chart-plot-area** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-plot-area** component to add **PlotArea** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartPlotArea** component is derived from the **PlotArea** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- column
- height
- initialized
- isInitialized
- name
- row
- style
- width
- wjProperty

Methods

- created

Constructor

constructor

```
constructor(options?: any): PlotArea
```

Initializes a new instance of the **PlotArea** class.

Parameters

- **options**: **any** OPTIONAL
  Initialization options for the plot area.

Inherited From
PlotArea
Returns
PlotArea

Properties

column

Gets or sets the column index of plot area. This determines the horizontal position of the plot area on the chart.

Inherited From
PlotArea
Type
number

height

Gets or sets the height of the plot area.

The height can be specified as a number (in pixels) or as a string in the format '{number}*' (star sizing).

Inherited From
PlotArea
Type
any

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

name

Gets or sets the plot area name.

Inherited From
`PlotArea`
Type
string

row

Gets or sets the row index of plot area. This determines the vertical position of the plot area on the chart.

Inherited From
`PlotArea`
Type
number

style

Gets or sets the style of the plot area.

Using **style** property, you can set appearance of the plot area. For example:

```
pa.style = { fill: 'rgba(0,255,0,0.1)' };
```

Inherited From
`PlotArea`
Type
any

width

Gets or sets width of the plot area.

The width can be specified as a number (in pixels) or as a string in the format '{number}*' (star sizing).

Inherited From
`PlotArea`
Type
any

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'plotAreas'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

WjFlexChartSeries Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
Series

Angular 2 component for the **Series** control.

The **wj-flex-chart-series** component must be contained in a **WjFlexChart** component.

Use the **wj-flex-chart-series** component to add **Series** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartSeries** component is derived from the **Series** control and inherits all its properties, events and methods.

The **wj-flex-chart-series** component may contain a **WjFlexChartAxis** child component.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SeriesBase
```

Initializes a new instance of the **SeriesBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
SeriesBase
Returns
SeriesBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart.

Inherited From
Series
Type
ChartType

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexPie Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
FlexPie

Angular 2 component for the **FlexPie** control.

Use the **wj-flex-pie** component to add **FlexPie** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexPie** component is derived from the **FlexPie** control and inherits all its properties, events and methods.

The **wj-flex-pie** component may contain the following child components: **WjFlexChartAnimation** , **WjFlexChartLegend** and **WjFlexPieDataLabel**.

Constructor

- constructor

Properties

- binding
- bindingName
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- innerRadius
- isAnimated
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- lostFocusNg
- offset
- palette
- plotMargin
- renderedNg
- renderingNg
- reversed
- rightToLeft
- selectedIndex
- selectedItemOffset
- selectedItemPosition
- selectionChangedNg
- selectionMode
- startAngle
- tooltip
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- pageToControl
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged

Constructor

constructor

```
constructor(element: any, options?): FlexPie
```

Initializes a new instance of the **FlexPie** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A Javascript object containing initialization data for the control.

Inherited From
FlexPie
Returns
FlexPie

Properties

binding

Gets or sets the name of the property that contains the chart values.

Inherited From
FlexPie
Type
string

bindingName

Gets or sets the name of the property that contains the name of the data items.

Inherited From
FlexPie
Type
string

collectionView

Gets the **ICollectionView** object that contains the chart data.

Inherited From
FlexChartBase
Type
ICollectionView

dataLabel

Gets or sets the point data label.

Inherited From
FlexPie
Type
PieDataLabel

footer

Gets or sets the text displayed in the chart footer.

Inherited From
FlexChartBase
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
FlexChartBase
Type
any

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

innerRadius

Gets or sets the size of the pie's inner radius.

The inner radius is measured as a fraction of the pie radius.

The default value for this property is zero, which creates a pie. Setting this property to values greater than zero creates pies with a hole in the middle, also known as doughnut charts.

Inherited From
`FlexPie`
Type
number

isAnimated

Gets or sets a value indicating whether to use animation when items are selected.

See also the `selectedItemPosition` and `selectionMode` properties.

Inherited From
`FlexPie`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the data used to create the chart.

Inherited From
`FlexChartBase`
Type
any

legend

Gets or sets the chart legend.

Inherited From
`FlexChartBase`
Type
`Legend`

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

offset

Gets or sets the offset of the slices from the pie center.

The offset is measured as a fraction of the pie radius.

Inherited From
`FlexPie`
Type
number

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **`Palettes`** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
`FlexChartBase`
Type
string[]

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

reversed

Gets or sets a value that determines whether angles are reversed (counter-clockwise).

The default value is false, which causes angles to be measured in the clockwise direction.

Inherited From
`FlexPie`
Type
boolean

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selectedIndex

Gets or sets the index of the selected slice.

Inherited From
`FlexPie`
Type
number

selectedItemOffset

Gets or sets the offset of the selected slice from the pie center.

Offsets are measured as a fraction of the pie radius.

Inherited From
`FlexPie`
Type
number

selectedItemPosition

Gets or sets the position of the selected slice.

Setting this property to a value other than 'None' causes the pie to rotate when an item is selected.

Note that in order to select slices by clicking the chart, you must set the **`selectionMode`** property to "Point".

Inherited From
`FlexPie`
Type
`Position`

selectionChangedNg

Angular (EventEmitter) version of the Wijmo **selectionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanged** Wijmo event name.

Type
EventEmitter

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

startAngle

Gets or sets the starting angle for the pie slices, in degrees.

Angles are measured clockwise, starting at the 9 o'clock position.

Inherited From
`FlexPie`
Type
number

tooltip

Gets the chart's **`Tooltip`**.

Inherited From
`FlexPie`
Type
`ChartTooltip`

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **`dispose`** method automatically removes any event listeners added with the **`addEventListener`** method.

Calling the **`dispose`** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
`Control`
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
Control
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexPie`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
FlexChartBase
Returns
Point

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

WjFlexPieDataLabel Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart
Base Class
PieDataLabel

Angular 2 component for the **PieDataLabel** control.

The **wj-flex-pie-data-label** component must be contained in a **WjFlexPie** component.

Use the **wj-flex-pie-data-label** component to add **PieDataLabel** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexPieDataLabel** component is derived from the **PieDataLabel** control and inherits all its properties, events and methods.

Properties

- border
- connectingLine
- content
- initialized
- isInitialized
- offset
- position
- renderingNg
- wjProperty

Methods

- created
- onRendering

Events

- rendering

Properties

border

Gets or sets a value indicating whether the data labels have borders.

Inherited From
`DataLabelBase`
Type
boolean

connectingLine

Gets or sets a value indicating whether to draw lines that connect labels to the data points.

Inherited From
`DataLabelBase`
Type
boolean

content

Gets or sets the content of data labels.

The content can be specified as a string or as a function that takes **HitTestInfo** object as a parameter.

When the label content is a string, it can contain any of the following parameters:

- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point.
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point.
- **propertyName**: any property of data object.

The parameter must be enclosed in curly brackets, for example 'x={x}, y={y}'.

In the following example, we show the y value of the data point in the labels.

```
// Create a chart and show y data in labels positioned above the data point.
 var chart = new wijmo.chart.FlexChart('#theChart');
 chart.initialize({
     itemsSource: data,
     bindingX: 'country',
     series: [
         { name: 'Sales', binding: 'sales' },
         { name: 'Expenses', binding: 'expenses' },
         { name: 'Downloads', binding: 'downloads' }],
 });
 chart.dataLabel.position = "Top";
 chart.dataLabel.content = "{country} {seriesName}:{y}";
```

The next example shows how to set data label content using a function.

```
// Set the data label content
 chart.dataLabel.content = function (ht) {
   return ht.name + ":" + ht.value.toFixed();
 }
```

Inherited From
DataLabelBase
Type
any

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

offset

Gets or sets the offset from label to the data point.

Inherited From
`DataLabelBase`
Type
number

position

Gets or sets the position of the data labels.

Inherited From
`PieDataLabel`
Type
`PieLabelPosition`

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'dataLabel'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

onRendering

```
onRendering(e: DataLabelRenderEventArgs): void
```

Raises the **`rendering`** event.

Parameters

- **e**: **DataLabelRenderEventArgs**
  The **`DataLabelRenderEventArgs`** object used to render the label.

Inherited From
`DataLabelBase`
Returns
void

Events

rendering

Occurs before the data label is rendered.

Inherited From
`DataLabelBase`
Arguments
`DataLabelRenderEventArgs`

wijmo/wijmo.angular2.chart.interaction Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.interaction

Contains Angular 2 components for the **wijmo.chart.interaction** module.

wijmo.angular2.chart.interaction is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjInteraction from 'wijmo/wijmo.angular2.chart.interaction';
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjInteraction.WjFlexChartRangeSelector, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-range-selector></wj-flex-chart-range-selector>
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFlexChartGestures
- WjFlexChartRangeSelector

WjFlexChartGestures Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.interaction
Base Class
ChartGestures

Angular 2 component for the **ChartGestures** control.

The **wj-flex-chart-gestures** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-gestures** component to add **ChartGestures** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartGestures** component is derived from the **ChartGestures** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- enable
- initialized
- interactiveAxes
- isInitialized
- mouseAction
- posX
- posY
- scaleX
- scaleY
- wjProperty

Methods

- created
- remove
- reset

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): ChartGestures
```

Initializes a new instance of the **ChartGestures** class.

Parameters

- **chart**: **FlexChartCore**
  The **FlexChart** that allows the user to zoom or pan.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Inherited From
ChartGestures
Returns
ChartGestures

Properties

enable

Gets or sets the enable of the ChartGestures.

Inherited From
ChartGestures
Type
boolean

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

- **interactiveAxes**

Gets or sets the interactive axes of the ChartGestures.

Inherited From
`ChartGestures`
Type
`InteractiveAxes`

- **isInitialized**

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

- **mouseAction**

Gets or sets the mouse action of the ChartGestures.

Inherited From
`ChartGestures`
Type
`MouseAction`

- **posX**

Gets or sets the initial position of the axis X. The value represents initial position on the axis when the Scale is less than 1. Otherwise, the Value has no effect. The Value should lie between 0 to 1.

Inherited From
`ChartGestures`
Type
number

- **posY**

Gets or sets the initial position of the axis Y. The value represents initial position on the axis when the Scale is less than 1. Otherwise, the Value has no effect. The Value should lie between 0 to 1.

Inherited From
`ChartGestures`
Type
number

- **scaleX**

Gets or sets the initial scale of axis X. The scale should be more than 0 and less than or equal to 1. The scale specifies which part of the range between Min and Max is shown. When scale is 1 (default value), the whole axis range is visible.

Inherited From
`ChartGestures`
Type
number

- **scaleY**

Gets or sets the initial scale of axis Y. The scale should be more than 0 and less than or equal to 1. The scale specifies which part of the range between Min and Max is shown. When scale is 1 (default value), the whole axis range is visible.

Inherited From
`ChartGestures`
Type
number

- **wjProperty**

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

remove

```
remove(): void
```

Removes the **ChartGestures** control from the chart.

Inherited From
ChartGestures
Returns
void

reset

```
reset(): void
```

Reset the axis of the chart.

Inherited From
ChartGestures
Returns
void

WjFlexChartRangeSelector Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.interaction
Base Class
RangeSelector

Angular 2 component for the **RangeSelector** control.

The **wj-flex-chart-range-selector** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-range-selector** component to add **RangeSelector** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartRangeSelector** component is derived from the **RangeSelector** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- initialized
- isInitialized
- isVisible
- max
- maxScale
- min
- minScale
- orientation
- rangeChangedNg
- seamless
- wjProperty

Methods

- created
- onRangeChanged
- remove

Events

- rangeChanged

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): RangeSelector
```

Initializes a new instance of the **RangeSelector** class.

Parameters

- **chart**: **FlexChartCore**
  The **FlexChart** that displays the selected range.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Inherited From
RangeSelector
Returns
RangeSelector

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the range selector.

Inherited From
RangeSelector
Type
boolean

max

Gets or sets the maximum value of the range. If not set, the maximum is calculated automatically.

Inherited From
`RangeSelector`
Type
number

maxScale

Gets or sets the maximum amount of data that can be selected, as a percentage of the total range. This property must be set to a value between zero and one.

Inherited From
`RangeSelector`
Type
number

min

Gets or sets the minimum value of the range. If not set, the minimum is calculated automatically.

Inherited From
`RangeSelector`
Type
number

minScale

Gets or sets the minimum amount of data that can be selected, as a percentage of the overall chart range. This property must be set to a value between zero and one.

Inherited From
`RangeSelector`
Type
number

orientation

Gets or sets the orientation of the range selector.

Inherited From
`RangeSelector`
Type
`Orientation`

rangeChangedNg

Angular (EventEmitter) version of the Wijmo **rangeChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rangeChanged** Wijmo event name.

Type
EventEmitter

seamless

Gets or sets a value that determines whether the min/max elements may be reversed by dragging one over the other.

Inherited From
`RangeSelector`
Type
boolean

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

onRangeChanged

```
onRangeChanged(e?: EventArgs): void
```

Raises the **rangeChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
RangeSelector
Returns
void

remove

```
remove(): void
```

Removes the **RangeSelector** control from the chart.

Inherited From
RangeSelector
Returns
void

Events

rangeChanged

Occurs after the range changes.

Inherited From
`RangeSelector`
Arguments
`EventArgs`

wijmo/wijmo.angular2.chart.animation Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.animation

Contains Angular 2 components for the **wijmo.chart.animation** module.

wijmo.angular2.chart.animation is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjAnimation from 'wijmo/wijmo.angular2.chart.animation';
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjAnimation.WjFlexChartAnimation, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-animation [animationMode]="'Point'"></wj-flex-chart-animation>
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFlexChartAnimation

WjFlexChartAnimation Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.animation
Base Class
ChartAnimation

Angular 2 component for the **ChartAnimation** control.

The **wj-flex-chart-animation** component must be contained in one of the following components: **WjFlexChart** , **WjFlexPie** , **WjFinancialChart** or **WjFlexRadar**.

Use the **wj-flex-chart-animation** component to add **ChartAnimation** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnimation** component is derived from the **ChartAnimation** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- animationMode
- axisAnimation
- duration
- easing
- initialized
- isInitialized
- wjProperty

Methods

- animate
- created

Constructor

constructor

```
constructor(chart: FlexChartBase, options?: any): ChartAnimation
```

Initializes a new instance of the **ChartAnimation** class.

Parameters

- **chart**: **FlexChartBase**
  A chart to which the **ChartAnimation** is attached.

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data for **ChartAnimation**.

Inherited From
ChartAnimation
Returns
ChartAnimation

Properties

animationMode

Gets or sets whether the plot points animate one at a time, series by series, or all at once. The whole animation is still completed within the duration.

Inherited From
ChartAnimation
Type
AnimationMode

axisAnimation

Gets or sets a value indicating whether animation is applied to the axis.

Inherited From
ChartAnimation
Type
boolean

- **duration**

Gets or sets the length of entire animation in milliseconds.

Inherited From
`ChartAnimation`
Type
number

- **easing**

Gets or sets the easing function applied to the animation.

Inherited From
`ChartAnimation`
Type
`Easing`

- **initialized**

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

- **isInitialized**

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

- **wjProperty**

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

Methods

animate

```
animate(): void
```

Performs the animation.

Inherited From
`ChartAnimation`
Returns
void

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

wijmo/wijmo.angular2.chart.analytics Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics

Contains Angular 2 components for the **wijmo.chart.analytics** module.

wijmo.angular2.chart.analytics is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjAnalitics from 'wijmo/wijmo.angular2.chart.analytics';
```

Classes

- WjFlexChartBoxWhisker
- WjFlexChartErrorBar
- WjFlexChartMovingAverage
- WjFlexChartParametricFunctionSeries
- WjFlexChartTrendLine
- WjFlexChartWaterfall
- WjFlexChartYFunctionSeries

WjFlexChartBoxWhisker Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
BoxWhisker

Angular 2 component for the **BoxWhisker** control.

The **wj-flex-chart-box-whisker** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-box-whisker** component to add **BoxWhisker** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartBoxWhisker** component is derived from the **BoxWhisker** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- gapWidth
- groupWidth
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- meanLineStyle
- meanMarkerStyle
- name
- quartileCalculation
- renderedNg
- renderingNg
- showInnerPoints
- showMeanLine
- showMeanMarker
- showOutliers
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): BoxWhisker
```

Initializes a new instance of the **BoxWhisker** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
BoxWhisker
Returns
BoxWhisker

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

gapWidth

Gets or sets a value that determines the width of the gab between groups as a percentage.

The default value for this property is 0.1. The min value is 0 and max value is 1.

Inherited From
BoxWhisker
Type
number

groupWidth

Gets or sets a value that determines the group width as a percentage.

The default value for this property is 0.8. The min value is 0 and max value is 1.

Inherited From
`BoxWhisker`
Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

meanLineStyle

Gets or sets a value that specifies the style for the mean line.

Inherited From
`BoxWhisker`
Type
any

meanMarkerStyle

Gets or sets a value that specifies the style for the mean marker.

Inherited From
`BoxWhisker`
Type
any

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

quartileCalculation

Gets or sets a value that specifies the quartile calculation method.

Inherited From
`BoxWhisker`
Type
`QuartileCalculation`

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

showInnerPoints

Gets or sets a value that determines whether to show the inner data points for each point in the series.

Inherited From
`BoxWhisker`
Type
boolean

showMeanLine

Gets or sets a value that determines whether to show the mean line.

Inherited From
`BoxWhisker`
Type
boolean

showMeanMarker

Gets or sets a value that determines whether to show the mean marker.

Inherited From
`BoxWhisker`
Type
boolean

showOutliers

Gets or sets a value that determines whether to show outliers.

Outliers are inner points outside the range between the first and third quartiles.

Inherited From
`BoxWhisker`
Type
boolean

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexChartErrorBar Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
ErrorBar

Angular 2 component for the **ErrorBar** control.

The **wj-flex-chart-error-bar** component must be contained in a **WjFlexChart** component.

Use the **wj-flex-chart-error-bar** component to add **ErrorBar** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartErrorBar** component is derived from the **ErrorBar** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- direction
- endStyle
- errorAmount
- errorBarStyle
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- value
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): ErrorBar
```

Initializes a new instance of the **ErrorBar** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
ErrorBar
Returns
ErrorBar

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart.

Inherited From
Series
Type
ChartType

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

direction

Gets or sets a value that specifies the direction of the error bars.

Inherited From
`ErrorBar`
Type
`ErrorBarDirection`

endStyle

Gets or sets a value that specifies the end style of the error bars.

Inherited From
`ErrorBar`
Type
`ErrorBarEndStyle`

errorAmount

Gets or sets a value that specifies the meaning of the **`value`** property.

Inherited From
`ErrorBar`
Type
`ErrorAmount`

errorBarStyle

Gets or sets the style used to render the error bars.

Inherited From
`ErrorBar`
Type
any

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

value

Gets or sets a value that specifies the error value of the series.

This property works with the **`errorAmount`** property.

Inherited From
`ErrorBar`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartMovingAverage Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
MovingAverage

Angular 2 component for the **MovingAverage** control.

The **wj-flex-chart-moving-average** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-moving-average** component to add **MovingAverage** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartMovingAverage** component is derived from the **MovingAverage** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility
- wjProperty

Methods

- approximate
- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): MovingAverage
```

Initializes a new instance of the **MovingAverage** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
MovingAverage
Returns
MovingAverage

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
SeriesBase
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period of the moving average series. It should be set to integer value greater than 1.

Inherited From
`MovingAverage`
Type
number

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

type

Gets or sets the type of the moving average series.

Inherited From
`MovingAverage`
Type
`MovingAverageType`

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Inherited From
`TrendLineBase`
Returns
number

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexChartParametricFunctionSeries Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
ParametricFunctionSeries

Angular 2 component for the **ParametricFunctionSeries** control.

The **wj-flex-chart-parametric-function-series** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-parametric-function-series** component to add **ParametricFunctionSeries** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartParametricFunctionSeries** component is derived from the **ParametricFunctionSeries** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- max
- min
- name
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty
- xFunc
- yFunc

Methods

- approximate
- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): ParametricFunctionSeries
```

Initializes a new instance of the **ParametricFunctionSeries** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
ParametricFunctionSeries
Returns
ParametricFunctionSeries

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
SeriesBase
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

max

Gets or sets the maximum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

min

Gets or sets the minimum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

xFunc

Gets or sets the function used to calculate the x value.

Inherited From
`ParametricFunctionSeries`
Type
Function

yFunc

Gets or sets the function used to calculate the y value.

Inherited From
`ParametricFunctionSeries`
Type
Function

Methods

approximate

```
approximate(value: number): void
```

Gets the approximate x and y from the given value.

Parameters

- **value**: **number**
  The value to calculate.

Inherited From
`ParametricFunctionSeries`
Returns
void

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartTrendLine Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
TrendLine

Angular 2 component for the **TrendLine** control.

The **wj-flex-chart-trend-line** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-trend-line** component to add **TrendLine** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartTrendLine** component is derived from the **TrendLine** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- coefficients
- collectionView
- cssClass
- fitType
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- order
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- approximate
- created
- dataToPoint
- drawLegendItem
- getDataRect
- getEquation
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): TrendLine
```

Initializes a new instance of the **TrendLine** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
TrendLine
Returns
TrendLine

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **`FlexChart`** object that owns this series.

Inherited From
`SeriesBase`
Type
`FlexChartCore`

coefficients

Gets the coefficients of the equation.

Inherited From
`TrendLine`
Type
number[]

collectionView

Gets the **`ICollectionView`** object that contains the data for this series.

Inherited From
`SeriesBase`
Type
`ICollectionView`

cssClass

Gets or sets the series CSS class.

Inherited From
`SeriesBase`
Type
string

fitType

Gets or sets the fit type of the **`TrendLine`**.

Inherited From
`TrendLine`
Type
`TrendLineFitType`

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or **`ICollectionView`** object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

order

Gets or sets the number of terms in a polynomial or Fourier equation.

Set this value to an integer greater than 1. It gets applied when the fitType is set to wijmo.chart.analytics.TrendLineFitType.Polynomial or wijmo.chart.analytics.TrendLineFitType.Fourier.

Inherited From
`TrendLine`
Type
number

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Inherited From
TrendLine
Returns
number

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getEquation

```
getEquation(fmt?: Function): void
```

Gets the formatted equation string for the coefficients.

Parameters

- **fmt**: **Function** OPTIONAL
  The formatting function used to convert the coefficients into strings. This parameter is optional.

Inherited From
`TrendLine`
Returns
void

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartWaterfall Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
Waterfall

Angular 2 component for the **Waterfall** control.

The **wj-flex-chart-waterfall** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-waterfall** component to add **Waterfall** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartWaterfall** component is derived from the **Waterfall** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- connectorLines
- cssClass
- hostElement
- initialized
- intermediateTotalLabels
- intermediateTotalPositions
- isInitialized
- itemsSource
- legendElement
- name
- relativeData
- renderedNg
- renderingNg
- showIntermediateTotal
- showTotal
- start
- startLabel
- style
- styles
- symbolMarker
- symbolSize
- symbolStyle
- totalLabel
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Waterfall
```

Initializes a new instance of the **Waterfall** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Waterfall
Returns
Waterfall

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

connectorLines

Gets or sets a value that determines whether to show connector lines.

Inherited From
Waterfall
Type
boolean

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

intermediateTotalLabels

Gets or sets the name of the property that contains labels for the intermediate total bars. This should be an array or a string.

This property works with the **`showIntermediateTotal`** and **`intermediateTotalPositions`** properties.

Inherited From
`Waterfall`
Type
any

intermediateTotalPositions

Gets or sets a value of the property that contains the index for positions of the intermediate total bars.

This property works with the **`showIntermediateTotal`** and **`intermediateTotalLabels`** properties.

Inherited From
`Waterfall`
Type
number[]

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or **ICollectionView** object that contains the series data.

Inherited From
SeriesBase
Type
any

legendElement

Gets the series element in the legend.

Inherited From
SeriesBase
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
SeriesBase
Type
string

relativeData

Gets or sets a value that determines whether the given data represents absolute or relative values (differences).

Inherited From
`Waterfall`
Type
boolean

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

showIntermediateTotal

Gets or sets a value that determines whether to show intermediate total bars.

This property works with **`intermediateTotalPositions`** and **`intermediateTotalLabels`** properties.

Inherited From
`Waterfall`
Type
boolean

showTotal

Gets or sets a value that determines whether to show the total bar at the end of the chart.

Inherited From
`Waterfall`
Type
boolean

start

Gets or sets a value that determines the value of the start bar. If start is null, the start bar will not be shown.

Inherited From
`Waterfall`
Type
number

startLabel

Gets or sets the label of the start bar.

Inherited From
`Waterfall`
Type
string

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

styles

Gets or sets the Waterfall styles.

The following styles are supported:

1. **start**: Specifies the style of the start column.
2. **total**: Specifies the style of the total column.
3. **intermediateTotal**: Specifies the style of the intermediate total column.
4. **falling**: Specifies the style of the falling columns.
5. **rising**: Specifies the style of the rising columns.
6. **connectorLines**: Specifies the style of the connectorLines.

```
waterfall.styles = {
  start: { fill: 'blue', stroke: 'blue' },
  total: { fill: 'yellow', stroke: 'yellow' },
  falling: { fill: 'red', stroke: 'red' },
  rising: { fill: 'green', stroke: 'green' },
  connectorLines: { stroke: 'blue', 'stroke-dasharray': '10, 10' }
}
```

Inherited From
`Waterfall`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

totalLabel

Gets or sets the label of the total bar.

Inherited From
`Waterfall`
Type
string

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexChartYFunctionSeries Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.analytics
Base Class
YFunctionSeries

Angular 2 component for the **YFunctionSeries** control.

The **wj-flex-chart-y-function-series** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-y-function-series** component to add **YFunctionSeries** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartYFunctionSeries** component is derived from the **YFunctionSeries** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- func
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- max
- min
- name
- renderedNg
- renderingNg
- sampleCount
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- approximate
- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): YFunctionSeries
```

Initializes a new instance of the **YFunctionSeries** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
YFunctionSeries
Returns
YFunctionSeries

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

func

Gets or sets the function used to calculate Y value.

Inherited From
YFunctionSeries
Type
Function

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

max

Gets or sets the maximum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

min

Gets or sets the minimum value of the parameter for calculating a function.

Inherited From
`FunctionSeries`
Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

sampleCount

Gets or sets the sample count for function calculation. The property doesn't apply for MovingAverage.

Inherited From
`TrendLineBase`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

approximate

```
approximate(x: number): number
```

Gets the approximate y value from the given x value.

Parameters

- **x**: **number**
  The x value to be used for calculating the Y value.

Inherited From
YFunctionSeries
Returns
number

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

wijmo/wijmo.angular2.chart.annotation Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation

Contains Angular 2 components for the **wijmo.chart.annotation** module.

wijmo.angular2.chart.annotation is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjAnnotation from 'wijmo/wijmo.angular2.chart.annotation';
import * as wjChart from 'wijmo/wijmo.angular2.chart';

@Component({
    directives: [wjChart.WjFlexChart, wjAnnotation.WjFlexChartAnnotationLayer,
           wjAnnotation.WjFlexChartAnnotationCircle, wjChart.WjFlexChartSeries],
    template: `
      <wj-flex-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-flex-chart-series [binding]="'y'"></wj-flex-chart-series>
          <wj-flex-chart-annotation-layer>
              <wj-flex-chart-annotation-circle [radius]="40" [point]="{x: 250, y: 150}"></wj-flex-chart-annotation-circle>
          </wj-flex-chart-annotation-layer>
      </wj-flex-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFlexChartAnnotationCircle
- WjFlexChartAnnotationEllipse
- WjFlexChartAnnotationImage
- WjFlexChartAnnotationLayer
- WjFlexChartAnnotationLine
- WjFlexChartAnnotationPolygon
- WjFlexChartAnnotationRectangle
- WjFlexChartAnnotationSquare
- WjFlexChartAnnotationText

WjFlexChartAnnotationCircle Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Circle

Angular 2 component for the **Circle** control.

The **wj-flex-chart-annotation-circle** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-circle** component to add **Circle** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationCircle** component is derived from the **Circle** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-circle** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- radius
- seriesIndex
- style
- tooltip
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Circle
```

Initializes a new instance of the **Circle** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Circle
Returns
Circle

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

radius

Gets or sets the radius of the **`Circle`** annotation.

Inherited From
`Circle`
Type
number

- **seriesIndex**

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

- **style**

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

- **tooltip**

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

- **wjProperty**

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
AnnotationBase
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

WjFlexChartAnnotationEllipse Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Ellipse

Angular 2 component for the **Ellipse** control.

The **wj-flex-chart-annotation-ellipse** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-ellipse** component to add **Ellipse** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationEllipse** component is derived from the **Ellipse** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-ellipse** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- height
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Ellipse
```

Initializes a new instance of the **Ellipse** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Ellipse
Returns
Ellipse

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

height

Gets or sets the height of the **`Ellipse`** annotation.

Inherited From
`Ellipse`
Type
number

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **point**.

Inherited From
AnnotationBase
Type
Point

point

Gets or sets the point of the annotation. The coordinates of points depends on the **attachment** property. See **AnnotationAttachment** for further description.

Inherited From
AnnotationBase
Type
DataPoint

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
AnnotationBase
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **point**.

Inherited From
AnnotationBase
Type
AnnotationPosition

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

width

Gets or sets the width of the **`Ellipse`** annotation.

Inherited From
`Ellipse`
Type
number

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

WjFlexChartAnnotationImage Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Image

Angular 2 component for the **Image** control.

The **wj-flex-chart-annotation-image** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-image** component to add **Image** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationImage** component is derived from the **Image** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-image** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- height
- href
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Image
```

Initializes a new instance of the **Image** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Image
Returns
Image

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

height

Gets or sets the height of the **Image** annotation.

Inherited From
Image
Type
number

href

Gets or sets the href of the **Image** annotation.

Inherited From
Image
Type
string

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
AnnotationBase
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

width

Gets or sets the width of the **Image** annotation.

Inherited From
Image
Type
number

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
AnnotationBase
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

WjFlexChartAnnotationLayer Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
AnnotationLayer

Angular 2 component for the **AnnotationLayer** control.

The **wj-flex-chart-annotation-layer** component must be contained in one of the following components: **WjFlexChart** or **WjFinancialChart**.

Use the **wj-flex-chart-annotation-layer** component to add **AnnotationLayer** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationLayer** component is derived from the **AnnotationLayer** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-layer** component may contain the following child components: **WjFlexChartAnnotationText** , **WjFlexChartAnnotationEllipse** , **WjFlexChartAnnotationRectangle** , **WjFlexChartAnnotationLine** , **WjFlexChartAnnotationPolygon** , **WjFlexChartAnnotationCircle** , **WjFlexChartAnnotationSquare** and **WjFlexChartAnnotationImage**.

Constructor

- constructor

Properties

- initialized
- isInitialized
- items
- wjProperty

Methods

- created
- getItem
- getItems

Constructor

constructor

```
constructor(chart: FlexChartCore, options?): AnnotationLayer
```

Initializes a new instance of the **AnnotationLayer** class.

Parameters

- **chart**: **FlexChartCore**
  A chart to which the **AnnotationLayer** is attached.

- **options**: OPTIONAL
  A JavaScript object containing initialization data for **AnnotationLayer**.

Inherited From
AnnotationLayer
Returns
AnnotationLayer

Properties

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

items

Gets the collection of annotation elements in the **`AnnotationLayer`**.

Inherited From
`AnnotationLayer`
Type
`ObservableArray`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is ''.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

getItem

```
getItem(name: string): AnnotationBase
```

Gets an annotation element by name in the **AnnotationLayer**.

Parameters

- **name**: **string**
  The annotation's name.

Inherited From
AnnotationLayer
Returns
AnnotationBase

getItems

```
getItems(name: string): Array
```

Gets the annotation elements by name in the **AnnotationLayer**.

Parameters

- **name**: **string**
  The annotations' name.

Inherited From
AnnotationLayer
Returns
Array

WjFlexChartAnnotationLine Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Line

Angular 2 component for the **Line** control.

The **wj-flex-chart-annotation-line** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-line** component to add **Line** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationLine** component is derived from the **Line** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-line** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- end
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- start
- style
- tooltip
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Line
```

Initializes a new instance of the **Line** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Line
Returns
Line

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

end

Gets or sets the end point of the Line annotation.

Inherited From
`Line`
Type
`DataPoint`

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **point**.

Inherited From
AnnotationBase
Type
Point

point

Gets or sets the point of the annotation. The coordinates of points depends on the **attachment** property. See **AnnotationAttachment** for further description.

Inherited From
AnnotationBase
Type
DataPoint

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
AnnotationBase
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **point**.

Inherited From
AnnotationBase
Type
AnnotationPosition

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

start

Gets or sets the start point of the `Line` annotation.

Inherited From
`Line`
Type
`DataPoint`

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

WjFlexChartAnnotationPolygon Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Polygon

Angular 2 component for the **Polygon** control.

The **wj-flex-chart-annotation-polygon** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-polygon** component to add **Polygon** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationPolygon** component is derived from the **Polygon** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-polygon** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- points
- position
- seriesIndex
- style
- tooltip
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Polygon
```

Initializes a new instance of the **Polygon** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Polygon
Returns
Polygon

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

points

Gets the collection of points of the **`Polygon`** annotation.

Inherited From
`Polygon`
Type
`ObservableArray`

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

- **seriesIndex**

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

- **style**

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

- **tooltip**

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

- **wjProperty**

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
AnnotationBase
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

WjFlexChartAnnotationRectangle Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Rectangle

Angular 2 component for the **Rectangle** control.

The **wj-flex-chart-annotation-rectangle** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-rectangle** component to add **Rectangle** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationRectangle** component is derived from the **Rectangle** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-rectangle** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- height
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- width
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Rectangle
```

Initializes a new instance of the **Rectangle** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Rectangle
Returns
Rectangle

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

- **height**

Gets or sets the height of the **`Rectangle`** annotation.

Inherited From
`Rectangle`
Type
number

- **initialized**

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

- **isInitialized**

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

- **isVisible**

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

- **name**

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **point**.

Inherited From
AnnotationBase
Type
Point

point

Gets or sets the point of the annotation. The coordinates of points depends on the **attachment** property. See **AnnotationAttachment** for further description.

Inherited From
AnnotationBase
Type
DataPoint

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
AnnotationBase
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **point**.

Inherited From
AnnotationBase
Type
AnnotationPosition

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

width

Gets or sets the width of the **`Rectangle`** annotation.

Inherited From
`Rectangle`
Type
number

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

WjFlexChartAnnotationSquare Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Square

Angular 2 component for the **Square** control.

The **wj-flex-chart-annotation-square** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-square** component to add **Square** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationSquare** component is derived from the **Square** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-square** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- content
- initialized
- isInitialized
- isVisible
- length
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- tooltip
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Square
```

Initializes a new instance of the **Square** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Square
Returns
Square

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

content

Gets or sets the text of the annotation.

Inherited From
Shape
Type
string

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

length

Gets or sets the length of the **`Square`** annotation.

Inherited From
`Square`
Type
number

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **point**.

Inherited From
AnnotationBase
Type
Point

point

Gets or sets the point of the annotation. The coordinates of points depends on the **attachment** property. See **AnnotationAttachment** for further description.

Inherited From
AnnotationBase
Type
DataPoint

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
AnnotationBase
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **point**.

Inherited From
AnnotationBase
Type
AnnotationPosition

- **seriesIndex**

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

- **style**

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

- **tooltip**

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

- **wjProperty**

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
AnnotationBase
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
AnnotationBase
Returns
void

WjFlexChartAnnotationText Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.annotation
Base Class
Text

Angular 2 component for the **Text** control.

The **wj-flex-chart-annotation-text** component must be contained in a **WjFlexChartAnnotationLayer** component.

Use the **wj-flex-chart-annotation-text** component to add **Text** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAnnotationText** component is derived from the **Text** control and inherits all its properties, events and methods.

The **wj-flex-chart-annotation-text** component may contain a **WjFlexChartDataPoint** child component.

Constructor

- constructor

Properties

- attachment
- initialized
- isInitialized
- isVisible
- name
- offset
- point
- pointIndex
- position
- seriesIndex
- style
- text
- tooltip
- wjProperty

Methods

- created
- destroy
- render

Constructor

constructor

```
constructor(options?: any): Text
```

Initializes a new instance of the **Text** annotation class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Text
Returns
Text

Properties

attachment

Gets or sets the attachment of the annotation.

Inherited From
AnnotationBase
Type
AnnotationAttachment

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isVisible

Gets or sets the visibility of the annotation.

Inherited From
`AnnotationBase`
Type
boolean

name

Gets or sets the name of the annotation.

Inherited From
`AnnotationBase`
Type
string

offset

Gets or sets the offset of the annotation from the **`point`**.

Inherited From
`AnnotationBase`
Type
`Point`

point

Gets or sets the point of the annotation. The coordinates of points depends on the **`attachment`** property. See **`AnnotationAttachment`** for further description.

Inherited From
`AnnotationBase`
Type
`DataPoint`

pointIndex

Gets or sets the data point index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

position

Gets or sets the position of the annotation. The position is relative to the **`point`**.

Inherited From
`AnnotationBase`
Type
`AnnotationPosition`

seriesIndex

Gets or sets the data series index of the annotation. Applies only when the **attachment** property is set to DataIndex.

Inherited From
`AnnotationBase`
Type
number

style

Gets or sets the style of the annotation.

Inherited From
`AnnotationBase`
Type
any

text

Gets or sets the text of the annotation.

Inherited From
`Text`
Type
string

tooltip

Gets or sets the tooltip of the annotation.

Inherited From
`AnnotationBase`
Type
string

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'items'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

destroy

```
destroy(): void
```

Destroy this annotation

Inherited From
`AnnotationBase`
Returns
void

render

```
render(engine: IRenderEngine): void
```

Render this annotation.

Parameters

- **engine**: **IRenderEngine**
  The engine to render annotation.

Inherited From
`AnnotationBase`
Returns
void

wijmo/wijmo.angular2.chart.finance Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance

Contains Angular 2 components for the **wijmo.chart.finance** module.

wijmo.angular2.chart.finance is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjFinance from 'wijmo/wijmo.angular2.chart.finance';

@Component({
    directives: [wjFinance.WjFinancialChart, wjFinance.WjFinancialChartSeries],
    template: `
      <wj-financial-chart [itemsSource]="data" [bindingX]="'x'">
          <wj-financial-chart-series [binding]="'y'"></wj-financial-chart-series>
      </wj-financial-chart>`,
    selector: 'my-cmp',
})
export class MyCmp {
    data: any[];
}
```

Classes

- WjFinancialChart
- WjFinancialChartSeries

WjFinancialChart Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance
Base Class
FinancialChart

Angular 2 component for the **FinancialChart** control.

Use the **wj-financial-chart** component to add **FinancialChart** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFinancialChart** component is derived from the **FinancialChart** control and inherits all its properties, events and methods.

The **wj-financial-chart** component may contain the following child components: **WjFlexChartTrendLine** , **WjFlexChartMovingAverage** , **WjFlexChartYFunctionSeries** , **WjFlexChartParametricFunctionSeries** , **WjFlexChartWaterfall** , **WjFlexChartBoxWhisker** , **WjFlexChartAnimation** , **WjFlexChartAnnotationLayer** , **WjFlexChartFibonacci** , **WjFlexChartFibonacciArcs** , **WjFlexChartFibonacciFans** , **WjFlexChartFibonacciTimeZones** , **WjFlexChartAtr** , **WjFlexChartCci** , **WjFlexChartRsi** , **WjFlexChartWilliamsR** , **WjFlexChartMacd** , **WjFlexChartMacdHistogram** , **WjFlexChartStochastic** , **WjFlexChartBollingerBands** , **WjFlexChartEnvelopes** , **WjFinancialChartSeries** , **WjFlexChartRangeSelector** , **WjFlexChartGestures** , **WjFlexChartAxis** , **WjFlexChartLegend** , **WjFlexChartLineMarker** and **WjFlexChartPlotArea**.

Constructor

- constructor

Properties

- asyncBindings
- axes
- axisX
- axisY
- binding
- bindingX
- chartType
- collectionView
- dataLabel
- footer
- footerStyle
- gotFocusNg
- header
- headerStyle
- hostElement
- initialized
- interpolateNulls
- isDisabled
- isInitialized
- isTouching
- isUpdating
- itemFormatter
- itemsSource
- legend
- legendToggle
- lostFocusNg
- options
- palette
- plotAreas
- plotMargin
- renderedNg
- renderingNg
- rightToLeft
- selection
- selectionChangedNg
- selectionMode
- series
- seriesVisibilityChangedNg
- symbolSize
- tooltip
- wjModelProperty

Methods

- addEventListener
- applyTemplate
- beginUpdate
- containsFocus
- created
- dataToPoint
- deferUpdate
- dispose
- disposeAll
- endUpdate
- focus
- getControl
- getTemplate
- hitTest
- initialize
- invalidate
- invalidateAll
- onGotFocus
- onLostFocus
- onRendered
- onRendering
- onSelectionChanged
- onSeriesVisibilityChanged
- pageToControl
- pointToData
- refresh
- refreshAll
- removeEventListener
- saveImageToDataUrl
- saveImageToFile

Events

- gotFocus
- lostFocus
- rendered
- rendering
- selectionChanged
- seriesVisibilityChanged

Constructor

constructor

```
constructor(element: any, options?): FinancialChart
```

Initializes a new instance of the **FlexChart** class.

Parameters

- **element**: **any**
  The DOM element that hosts the control, or a selector for the host element (e.g. '#theCtrl').

- **options**: OPTIONAL
  A JavaScript object containing initialization data for the control.

Inherited From
FinancialChart
Returns
FinancialChart

Properties

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axes

Gets the collection of **Axis** objects.

Inherited From
FlexChartCore
Type
ObservableArray

axisX

Gets or sets the main X axis.

Inherited From
`FlexChartCore`
Type
`Axis`

axisY

Gets or sets the main Y axis.

Inherited From
`FlexChartCore`
Type
`Axis`

binding

Gets or sets the name of the property that contains the Y values.

Inherited From
`FlexChartCore`
Type
string

bindingX

Gets or sets the name of the property that contains the X data values.

Inherited From
`FlexChartCore`
Type
string

chartType

Gets or sets the type of financial chart to create.

Inherited From
`FinancialChart`
Type
`FinancialChartType`

collectionView

Gets the `ICollectionView` object that contains the chart data.

Inherited From
`FlexChartBase`
Type
`ICollectionView`

dataLabel

Gets or sets the point data label.

Inherited From
`FlexChartCore`
Type
`DataLabel`

footer

Gets or sets the text displayed in the chart footer.

Inherited From
`FlexChartBase`
Type
string

footerStyle

Gets or sets the style of the chart footer.

Inherited From
`FlexChartBase`
Type
any

gotFocusNg

Angular (EventEmitter) version of the Wijmo **gotFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **gotFocus** Wijmo event name.

Type
EventEmitter

header

Gets or sets the text displayed in the chart header.

Inherited From
`FlexChartBase`
Type
string

headerStyle

Gets or sets the style of the chart header.

Inherited From
`FlexChartBase`
Type
any

hostElement

Gets the DOM element that is hosting the control.

Inherited From
`Control`
Type
HTMLElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

interpolateNulls

Gets or sets a value that determines whether to interpolate null values in the data.

If true, the chart interpolates the value of any missing data based on neighboring points. If false, it leaves a break in lines and areas at the points with null values.

Inherited From
`FlexChartCore`
Type
boolean

isDisabled

Gets or sets a value that determines whether the control is disabled.

Disabled controls cannot get mouse or keyboard events.

Inherited From
`Control`
Type
boolean

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

isTouching

Gets a value that indicates whether the control is currently handling a touch event.

Inherited From
`Control`
Type
boolean

isUpdating

Gets a value that indicates whether the control is currently being updated.

Inherited From
`Control`
Type
boolean

itemFormatter

Gets or sets the item formatter function that allows you to customize the appearance of data points. See the Explorer sample's **`Item Formatter (http://demos.wijmo.com/5/Angular/Explorer/Explorer/#/chart/itemFormatter)`** for a demonstration.

Inherited From
`FlexChartBase`
Type
Function

itemsSource

Gets or sets the array or **ICollectionView** object that contains the data used to create the chart.

Inherited From
FlexChartBase
Type
any

legend

Gets or sets the chart legend.

Inherited From
FlexChartBase
Type
Legend

legendToggle

Gets or sets a value indicating whether clicking legend items toggles the series visibility in the chart.

Inherited From
FlexChartCore
Type
boolean

lostFocusNg

Angular (EventEmitter) version of the Wijmo **lostFocus** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **lostFocus** Wijmo event name.

Type
EventEmitter

options

Gets or sets various chart options.

The following options are supported:

kagi.fields: Specifies the **`DataFields`** used for the Kagi chart. The default value is DataFields.Close.

kagi.rangeMode: Specifies the **`RangeMode`** for the Kagi chart. The default value is RangeMode.Fixed.

kagi.reversalAmount: Specifies the reversal amount for the Kagi chart. The default value is 14.

```
chart.options = {
  kagi: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     reversalAmount: 14
  }
}
```

lineBreak.newLineBreaks: Gets or sets the number of previous boxes that must be compared before a new box is drawn in Line Break charts. The default value is 3.

```
chart.options = {
  lineBreak: { newLineBreaks: 3 }
}
```

renko.fields: Specifies the **`DataFields`** used for the Renko chart. The default value is DataFields.Close.

renko.rangeMode: Specifies the **`RangeMode`** for the Renko chart. The default value is RangeMode.Fixed.

renko.boxSize: Specifies the box size for the Renko chart. The default value is 14.

```
chart.options = {
  renko: {
     fields: wijmo.chart.finance.DataFields.Close,
     rangeMode: wijmo.chart.finance.RangeMode.Fixed,
     boxSize: 14
  }
}
```

Inherited From
`FinancialChart`
Type
any

palette

Gets or sets an array of default colors to use for displaying each series.

The array contains strings that represents CSS colors. For example:

```
// use colors specified by name
chart.palette = ['red', 'green', 'blue'];

// or use colors specified as rgba-values
chart.palette = [
  'rgba(255,0,0,1)',
  'rgba(255,0,0,0.8)',
  'rgba(255,0,0,0.6)',
  'rgba(255,0,0,0.4)'];
```

There is a set of predefined palettes in the **Palettes** class that you can use, for example:

```
chart.palette = wijmo.chart.Palettes.coral;
```

Inherited From
FlexChartBase
Type
string[]

plotAreas

Gets the collection of **PlotArea** objects.

Inherited From
FlexChartCore
Type
PlotAreaCollection

plotMargin

Gets or sets the plot margin in pixels.

The plot margin represents the area between the edges of the control and the plot area.

By default, this value is calculated automatically based on the space required by the axis labels, but you can override it if you want to control the precise position of the plot area within the control (for example, when aligning multiple chart controls on a page).

You may set this property to a numeric value or to a CSS-style margin specification. For example:

```
// set the plot margin to 20 pixels on all sides
chart.plotMargin = 20;

// set the plot margin for top, right, bottom, left sides
chart.plotMargin = '10 15 20 25';

// set the plot margin for top/bottom (10px) and left/right (20px)
chart.plotMargin = '10 20';
```

Inherited From
`FlexChartBase`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

rightToLeft

Gets a value indicating whether the control is hosted in an element with right-to-left layout.

Inherited From
`Control`
Type
boolean

selection

Gets or sets the selected chart series.

Inherited From
`FlexChartCore`
Type
`SeriesBase`

selectionChangedNg

Angular (EventEmitter) version of the Wijmo **selectionChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **selectionChanged** Wijmo event name.

Type
EventEmitter

selectionMode

Gets or sets an enumerated value indicating whether or what is selected when the user clicks the chart.

Inherited From
`FlexChartBase`
Type
`SelectionMode`

series

Gets the collection of **`Series`** objects.

Inherited From
`FlexChartCore`
Type
`ObservableArray`

seriesVisibilityChangedNg

Angular (EventEmitter) version of the Wijmo **seriesVisibilityChanged** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **seriesVisibilityChanged** Wijmo event name.

Type
EventEmitter

symbolSize

Gets or sets the size of the symbols used for all Series objects in this **`FlexChart`**.

This property may be overridden by the symbolSize property on each **`Series`** object.

Inherited From
`FlexChartCore`
Type
number

tooltip

Gets the chart **Tooltip** object.

The tooltip content is generated using a template that may contain any of the following parameters:

- **propertyName**: Any property of the data object represented by the point.
- **seriesName**: Name of the series that contains the data point (FlexChart only).
- **pointIndex**: Index of the data point.
- **value**: **Value** of the data point (y-value for **FlexChart**, item value for **FlexPie**).
- **x**: **x**-value of the data point (FlexChart only).
- **y**: **y**-value of the data point (FlexChart only).
- **name**: **Name** of the data point (x-value for **FlexChart** or legend entry for **FlexPie**).

To modify the template, assign a new value to the tooltip's content property. For example:

```
chart.tooltip.content = '<b>{seriesName}</b> ' +
   '<img src="resources/{x}.png"/><br/>{y}';
```

You can disable chart tooltips by setting the template to an empty string.

You can also use the **tooltip** property to customize tooltip parameters such as **showDelay** and **hideDelay**:

```
chart.tooltip.showDelay = 1000;
```

See **ChartTooltip** properties for more details and options.

Inherited From
FlexChartCore
Type
ChartTooltip

wjModelProperty

Defines a name of a property represented by [(ngModel)] directive (if specified). Default value is ''.

Type
string

Methods

addEventListener

```
addEventListener(target: EventTarget, type: string, fn: any, capture?: boolean): void
```

Adds an event listener to an element owned by this **`Control`**.

The control keeps a list of attached listeners and their handlers, making it easier to remove them when the control is disposed (see the **`dispose`** and **`removeEventListener`** methods).

Failing to remove event listeners may cause memory leaks.

Parameters

- **target**: **EventTarget**
  Target element for the event.
- **type**: **string**
  String that specifies the event.
- **fn**: **any**
  Function to execute when the event occurs.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing.

Inherited From
`Control`
Returns
void

applyTemplate

```
applyTemplate(classNames: string, template: string, parts: Object, namePart?: string): HTMLElement
```

Applies the template to a new instance of a control, and returns the root element.

This method should be called by constructors of templated controls. It is responsible for binding the template parts to the corresponding control members.

For example, the code below applies a template to an instance of an **`InputNumber`** control. The template must contain elements with the 'wj-part' attribute set to 'input', 'btn-inc', and 'btn-dec'. The control members '_tbx', '_btnUp', and '_btnDn' will be assigned references to these elements.

```
this.applyTemplate('wj-control wj-inputnumber', template, {
  _tbx: 'input',
  _btnUp: 'btn-inc',
  _btnDn: 'btn-dec'
}, 'input');
```

Parameters

- **classNames**: **string**
  Names of classes to add to the control's host element.
- **template**: **string**
  An HTML string that defines the control template.
- **parts**: **Object**
  A dictionary of part variables and their names.
- **namePart**: **string** OPTIONAL
  Name of the part to be named after the host element. This determines how the control submits data when used in forms.

Inherited From
`Control`
Returns
HTMLElement

beginUpdate

```
beginUpdate(): void
```

Suspends notifications until the next call to **endUpdate**.

Inherited From
Control
Returns
void

containsFocus

```
containsFocus(): boolean
```

Checks whether this control contains the focused element.

Inherited From
Control
Returns
boolean

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: any, y?: number): Point
```

Converts a **`Point`** from data coordinates to control coordinates.

Parameters

- **pt**: **any**
  `Point` in data coordinates, or X coordinate of a point in data coordinates.

- **y**: **number** OPTIONAL
  Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

deferUpdate

```
deferUpdate(fn: Function): void
```

Executes a function within a **`beginUpdate`**/**`endUpdate`** block.

The control will not be updated until the function has been executed. This method ensures **`endUpdate`** is called even if the function throws an exception.

Parameters

- **fn**: **Function**
  Function to be executed.

Inherited From
`Control`
Returns
void

dispose

```
dispose(): void
```

Disposes of the control by removing its association with the host element.

The **dispose** method automatically removes any event listeners added with the **addEventListener** method.

Calling the **dispose** method is important in applications that create and remove controls dynamically. Failing to dispose of the controls may cause memory leaks.

Inherited From
Control
Returns
void

STATIC disposeAll

```
disposeAll(e?: HTMLElement): void
```

Disposes of all Wijmo controls contained in an HTML element.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element.

Inherited From
Control
Returns
void

endUpdate

```
endUpdate(): void
```

Resumes notifications suspended by calls to **beginUpdate**.

Inherited From
Control
Returns
void

focus

```
focus(): void
```

Sets the focus to this control.

Inherited From
`Control`
Returns
void

STATIC getControl

```
getControl(element: any): Control
```

Gets the control that is hosted in a given DOM element.

Parameters

- **element**: **any**
  The DOM element that is hosting the control, or a selector for the host element (e.g. '#theCtrl').

Inherited From
`Control`
Returns
`Control`

getTemplate

```
getTemplate(): string
```

Gets the HTML template used to create instances of the control.

This method traverses up the class hierarchy to find the nearest ancestor that specifies a control template. For example, if you specify a prototype for the **`ComboBox`** control, it will override the template defined by the **`DropDown`** base class.

Inherited From
`Control`
Returns
string

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the control by copying the properties from a given object.

This method allows you to initialize controls using plain data objects instead of setting the value of each property in code.

For example:

```
grid.initialize({
  itemsSource: myList,
  autoGenerateColumns: false,
  columns: [
    { binding: 'id', header: 'Code', width: 130 },
    { binding: 'name', header: 'Name', width: 60 }
  ]
});

// is equivalent to
grid.itemsSource = myList;
grid.autoGenerateColumns = false;

// etc.
```

The initialization data is type-checked as it is applied. If the initialization object contains unknown property names or invalid data types, this method will throw.

Parameters

- **options**: **any**
  Object that contains the initialization data.

Inherited From
`Control`
Returns
void

invalidate

```
invalidate(fullUpdate?: boolean): void
```

Invalidates the control causing an asynchronous refresh.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  Whether to update the control layout as well as the content.

Inherited From
Control
Returns
void

STATIC invalidateAll

```
invalidateAll(e?: HTMLElement): void
```

Invalidates all Wijmo controls contained in an HTML element.

Use this method when your application has dynamic panels that change the control's visibility or dimensions. For example, splitters, accordions, and tab controls usually change the visibility of its content elements. In this case, failing to notify the controls contained in the element may cause them to stop working properly.

If this happens, you must handle the appropriate event in the dynamic container and call the **invalidateAll** method so the contained Wijmo controls will update their layout information properly.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
Control
Returns
void

onGotFocus

```
onGotFocus(e?: EventArgs): void
```

Raises the **gotFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onLostFocus

```
onLostFocus(e?: EventArgs): void
```

Raises the **lostFocus** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
Control
Returns
void

onRendered

```
onRendered(e: RenderEventArgs): void
```

Raises the **rendered** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onRendering

```
onRendering(e: RenderEventArgs): void
```

Raises the **rendering** event.

Parameters

- **e**: **RenderEventArgs**
  The **RenderEventArgs** object used to render the chart.

Inherited From
FlexChartBase
Returns
void

onSelectionChanged

```
onSelectionChanged(e?: EventArgs): void
```

Raises the **selectionChanged** event.

Parameters

- **e**: **EventArgs** OPTIONAL

Inherited From
FlexChartBase
Returns
void

onSeriesVisibilityChanged

```
onSeriesVisibilityChanged(e: SeriesEventArgs): void
```

Raises the **seriesVisibilityChanged** event.

Parameters

- **e**: **SeriesEventArgs**
  The **SeriesEventArgs** object that contains the event data.

Inherited From
FlexChartCore
Returns
void

pageToControl

```
pageToControl(pt: any, y?: number): Point
```

Converts page coordinates to control coordinates.

Parameters

- **pt**: **any**
  The point of page coordinates or x value of page coordinates.

- **y**: **number** OPTIONAL
  The y value of page coordinates. Its value should be a number, if pt is a number type. However, the y parameter is optional when pt is Point type.

Inherited From
`FlexChartBase`
Returns
`Point`

pointToData

```
pointToData(pt: any, y?: number): Point
```

Converts a **`Point`** from control coordinates to chart data coordinates.

Parameters

- **pt**: **any**
  The point to convert, in control coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`FlexChartCore`
Returns
`Point`

refresh

```
refresh(fullUpdate?: boolean): void
```

Refreshes the chart.

Parameters

- **fullUpdate**: **boolean** OPTIONAL
  A value indicating whether to update the control layout as well as the content.

Inherited From
`FlexChartBase`
Returns
void

STATIC refreshAll

```
refreshAll(e?: HTMLElement): void
```

Refreshes all Wijmo controls contained in an HTML element.

This method is similar to **`invalidateAll`**, except the controls are updated immediately rather than after an interval.

Parameters

- **e**: **HTMLElement** OPTIONAL
  Container element. If set to null, all Wijmo controls on the page will be invalidated.

Inherited From
`Control`
Returns
void

removeEventListener

```
removeEventListener(target?: EventTarget, type?: string, fn?: any, capture?: boolean): number
```

Removes one or more event listeners attached to elements owned by this **Control**.

Parameters

- **target**: **EventTarget** OPTIONAL
  Target element for the event. If null, removes listeners attached to all targets.
- **type**: **string** OPTIONAL
  String that specifies the event. If null, removes listeners attached to all events.
- **fn**: **any** OPTIONAL
  Handler to remove. If null, removes all handlers.
- **capture**: **boolean** OPTIONAL
  Whether the listener is capturing. If null, removes capturing and non-capturing listeners.

Inherited From
Control
Returns
number

saveImageToDataUrl

```
saveImageToDataUrl(format: ImageFormat, done: Function): void
```

Save chart to image data url. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **format**: **ImageFormat**
  The **ImageFormat** for the exported image.
- **done**: **Function**
  A function to be called after data url is generated. The function gets passed the data url as its argument.

Inherited From
FlexChartBase
Returns
void

saveImageToFile

```
saveImageToFile(filename: string): void
```

Save chart to an image file. The function doesn't work in IE browsers. Add wijmo.chart.render module on page to support chart export in IE browsers.

Parameters

- **filename**: **string**
  The filename for the exported image file including extension. Supported types are PNG, JPEG and SVG.

Inherited From
`FlexChartBase`
Returns
void

Events

gotFocus

Occurs when the control gets the focus.

Inherited From
`Control`
Arguments
`EventArgs`

lostFocus

Occurs when the control loses the focus.

Inherited From
`Control`
Arguments
`EventArgs`

rendered

Occurs after the chart finishes rendering.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

rendering

Occurs before the chart starts rendering data.

Inherited From
`FlexChartBase`
Arguments
`RenderEventArgs`

selectionChanged

Occurs after the selection changes, whether programmatically or when the user clicks the chart. This is useful, for example, when you want to update details in a textbox showing the current selection.

Inherited From
`FlexChartBase`
Arguments
`EventArgs`

seriesVisibilityChanged

Occurs when the series visibility changes, for example when the legendToggle property is set to true and the user clicks the legend.

Inherited From
`FlexChartCore`
Arguments
`SeriesEventArgs`

WjFinancialChartSeries Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance
Base Class
FinancialSeries

Angular 2 component for the **FinancialSeries** control.

The **wj-financial-chart-series** component must be contained in a **WjFinancialChart** component.

Use the **wj-financial-chart-series** component to add **FinancialSeries** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFinancialChartSeries** component is derived from the **FinancialSeries** control and inherits all its properties, events and methods.

The **wj-financial-chart-series** component may contain a **WjFlexChartAxis** child component.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- chartType
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): SeriesBase
```

Initializes a new instance of the **SeriesBase** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
SeriesBase
Returns
SeriesBase

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

chartType

Gets or sets the chart type for a specific series, overriding the chart type set on the overall chart. Please note that ColumnVolume, EquiVolume, CandleVolume and ArmsCandleVolume chart types are not supported and should be set on the **FinancialChart**.

Inherited From
FinancialSeries
Type
FinancialChartType

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

wijmo/wijmo.angular2.chart.finance.analytics Module

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics

Contains Angular 2 components for the **wijmo.chart.finance.analytics** module.

wijmo.angular2.chart.finance.analytics is an external TypeScript module that can be imported to your code using its ambient module name. For example:

```
import * as wjFinanceAnalitics from 'wijmo/wijmo.angular2.chart.finance.analytics';
```

Classes

- WjFlexChartAtr
- WjFlexChartBollingerBands
- WjFlexChartCci
- WjFlexChartEnvelopes
- WjFlexChartFibonacci
- WjFlexChartFibonacciArcs
- WjFlexChartFibonacciFans
- WjFlexChartFibonacciTimeZones
- WjFlexChartMacd
- WjFlexChartMacdHistogram
- WjFlexChartRsi
- WjFlexChartStochastic
- WjFlexChartWilliamsR

WjFlexChartAtr Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
ATR

Angular 2 component for the **ATR** control.

The **wj-flex-chart-atr** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-atr** component to add **ATR** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartAtr** component is derived from the **ATR** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): ATR
```

Initializes a new instance of the **ATR** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
ATR
Returns
ATR

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
SeriesBase
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`SingleOverlayIndicatorBase`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexChartBollingerBands Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
BollingerBands

Angular 2 component for the **BollingerBands** control.

The **wj-flex-chart-bollinger-bands** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-bollinger-bands** component to add **BollingerBands** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartBollingerBands** component is derived from the **BollingerBands** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- multiplier
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): BollingerBands
```

Initializes a new instance of the **BollingerBands** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
BollingerBands
Returns
BollingerBands

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
SeriesBase
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

multiplier

Gets or sets the standard deviation multiplier.

Inherited From
`BollingerBands`
Type
number

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`BollingerBands`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartCci Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
CCI

Angular 2 component for the **CCI** control.

The **wj-flex-chart-cci** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-cci** component to add **CCI** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartCci** component is derived from the **CCI** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- constant
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): CCI
```

Initializes a new instance of the **CCI** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
CCI
Returns
CCI

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

constant

Gets or sets the constant value for the CCI calculation. The default value is 0.015.

Inherited From
CCI
Type
number

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`SingleOverlayIndicatorBase`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartEnvelopes Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
Envelopes

Angular 2 component for the **Envelopes** control.

The **wj-flex-chart-envelopes** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-envelopes** component to add **Envelopes** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartEnvelopes** component is derived from the **Envelopes** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- hostElement
- initialized
- isInitialized
- itemsSource
- legendElement
- name
- period
- renderedNg
- renderingNg
- size
- style
- symbolMarker
- symbolSize
- symbolStyle
- type
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Envelopes
```

Initializes a new instance of the **Envelopes** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Envelopes
Returns
Envelopes

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

hostElement

Gets the series host element.

Inherited From
SeriesBase
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

period

Gets or sets the period for the calculation as an integer value.

Inherited From
`Envelopes`
Type
any

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

size

Gets or set the size of the moving average envelopes. The default value is 2.5 percent (0.025).

Inherited From
`Envelopes`
Type
number

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

type

Gets or sets the moving average type for the envelopes. The default value is Simple.

Inherited From
`Envelopes`
Type
`MovingAverageType`

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartFibonacci Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
Fibonacci

Angular 2 component for the **Fibonacci** control.

The **wj-flex-chart-fibonacci** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-fibonacci** component to add **Fibonacci** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartFibonacci** component is derived from the **Fibonacci** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- high
- hostElement
- initialized
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- low
- maxX
- minX
- name
- renderedNg
- renderingNg
- style
- symbolMarker
- symbolSize
- symbolStyle
- uptrend
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): Fibonacci
```

Initializes a new instance of the **Fibonacci** class.

Parameters

- **options**: **any** OPTIONAL
  JavaScript object containing initialization data for the object.

Inherited From
Fibonacci
Returns
Fibonacci

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

high

Gets or sets the high value of **Fibonacci** tool.

If not specified, the high value is caclulated based on data values provided by the **itemsSource**.

Inherited From
Fibonacci
Type
number

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

labelPosition

Gets or sets the label position for levels in `Fibonacci` tool.

Inherited From
`Fibonacci`
Type
`LabelPosition`

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [0, 23.6, 38.2, 50, 61.8, 100].

Inherited From
`Fibonacci`
Type
number[]

low

Gets or sets the low value of `Fibonacci` tool.

If not specified, the low value is calculated based on data values provided by **itemsSource**.

Inherited From
`Fibonacci`
Type
number

maxX

Gets or sets the x maximum value of the `Fibonacci` tool.

If not specified, current maximum of x-axis is used. The value can be specified as a number or Date object.

Inherited From
`Fibonacci`
Type
any

minX

Gets or sets the x minimal value of the **`Fibonacci`** tool.

If not specified, current minimum of x-axis is used. The value can be specified as a number or Date object.

Inherited From
`Fibonacci`
Type
any

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

style

Gets or sets the series style.

Inherited From
`SeriesBase`
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
`Marker`

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

uptrend

Gets or sets a value indicating whether to create uptrending **`Fibonacci`** tool.

Default value is true(uptrend). If the value is false, the downtrending levels are plotted.

Inherited From
`Fibonacci`
Type
boolean

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **`Point`** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  `Point` in series data coordinates.

Inherited From
`SeriesBase`
Returns
`Point`

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **rect**: **Rect**
  The position of the legend item.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.

- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
`SeriesBase`
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **`HitTestInfo`** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
`SeriesBase`
Returns
`HitTestInfo`

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
`SeriesBase`
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
`SeriesBase`
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
`Size`

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

- **index**: **number**
  The index of the series to render.

- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
SeriesBase
Arguments
IRenderEngine

rendering

Occurs when series is rendering.

Inherited From
SeriesBase
Arguments
EventArgs

WjFlexChartFibonacciArcs Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
FibonacciArcs

Angular 2 component for the **FibonacciArcs** control.

The **wj-flex-chart-fibonacci-arcs** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-fibonacci-arcs** component to add **FibonacciArcs** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartFibonacciArcs** component is derived from the **FibonacciArcs** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- initialized
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- name
- renderedNg
- renderingNg
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FibonacciArcs
```

Initializes a new instance of the **FibonacciArcs** class.

Parameters

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data.

Inherited From
FibonacciArcs
Returns
FibonacciArcs

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

end

Gets or sets the ending **DataPoint** for the base line.

The **DataPoint** x value can be a number or a Date object (for time-based data).

Unlike some of the other Fibonacci tools, the ending **DataPoint** is **not** calculated automatically if undefined.

Inherited From
FibonacciArcs
Type
DataPoint

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

labelPosition

Gets or sets the `LabelPosition` for levels in `FibonacciArcs` tool.

Inherited From
`FibonacciArcs`
Type
`LabelPosition`

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [38.2, 50, 61.8].

Inherited From
`FibonacciArcs`
Type
number[]

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

start

Gets or sets the starting **DataPoint** for the base line.

The **DataPoint** x value can be a number or a Date object (for time-based data).

Unlike some of the other Fibonacci tools, the starting **DataPoint** is **not** calculated automatically if undefined.

Inherited From
FibonacciArcs
Type
DataPoint

style

Gets or sets the series style.

Inherited From
SeriesBase
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
SeriesBase
Type
Marker

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexChartFibonacciFans Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
FibonacciFans

Angular 2 component for the **FibonacciFans** control.

The **wj-flex-chart-fibonacci-fans** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-fibonacci-fans** component to add **FibonacciFans** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartFibonacciFans** component is derived from the **FibonacciFans** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- end
- hostElement
- initialized
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- name
- renderedNg
- renderingNg
- start
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FibonacciFans
```

Initializes a new instance of the **FibonacciFans** class.

Parameters

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data.

Inherited From
FibonacciFans
Returns
FibonacciFans

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

end

Gets or sets the ending **DataPoint** for the base line.

If not set, the starting **DataPoint** is calculated automatically. The **DataPoint** x value can be a number or a Date object (for time-based data).

Inherited From
FibonacciFans
Type
DataPoint

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

labelPosition

Gets or sets the `LabelPosition` for levels in `FibonacciFans` tool.

Inherited From
`FibonacciFans`
Type
`LabelPosition`

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [0, 23.6, 38.2, 50, 61.8, 100].

Inherited From
`FibonacciFans`
Type
number[]

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter

renderingNg

Angular (EventEmitter) version of the Wijmo **rendering** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendering** Wijmo event name.

Type
EventEmitter

start

Gets or sets the starting **DataPoint** for the base line.

If not set, the starting **DataPoint** is calculated automatically. The **DataPoint** x value can be a number or a Date object (for time-based data).

Inherited From
FibonacciFans
Type
DataPoint

style

Gets or sets the series style.

Inherited From
SeriesBase
Type
any

symbolMarker

Gets or sets the shape of marker to use for each data point in the series. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
SeriesBase
Type
Marker

symbolSize

Gets or sets the size (in pixels) of the symbols used to render this **`Series`**. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
number

symbolStyle

Gets or sets the series symbol style. Applies to Scatter, LineSymbols, and SplineSymbols chart types.

Inherited From
`SeriesBase`
Type
any

visibility

Gets or sets an enumerated value indicating whether and where the series appears.

Inherited From
`SeriesBase`
Type
`SeriesVisibility`

wjProperty

Gets or sets a name of a property that this component is assigned to. Default value is 'series'.

Type
string

Methods

created

```
created(): void
```

If you create a custom component inherited from a Wijmo component, you can override this method and perform necessary initializations that you usually do in a class constructor. This method is called in the last line of a Wijmo component constructor and allows you to not declare your custom component's constructor at all, thus preventing you from a necessity to maintain constructor parameters and keep them in synch with Wijmo component's constructor parameters.

Returns
void

dataToPoint

```
dataToPoint(pt: Point): Point
```

Converts a **Point** from series data coordinates to control coordinates.

Parameters

- **pt**: **Point**
  Point in series data coordinates.

Inherited From
SeriesBase
Returns
Point

drawLegendItem

```
drawLegendItem(engine: IRenderEngine, rect: Rect, index: number): void
```

Draw a legend item at the specified position.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.
- **rect**: **Rect**
  The position of the legend item.
- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
`SeriesBase`
Returns
void

getDataRect

```
getDataRect(currentRect?: Rect, calculatedRect?: Rect): Rect
```

Returns the series bounding rectangle in data coordinates.

If getDataRect() returns null, the limits are calculated automatically based on the data values.

Parameters

- **currentRect**: **Rect** OPTIONAL
  The current rectangle of chart. This parameter is optional.
- **calculatedRect**: **Rect** OPTIONAL
  The calculated rectangle of chart. This parameter is optional.

Inherited From
`SeriesBase`
Returns
`Rect`

getPlotElement

```
getPlotElement(pointIndex: number): any
```

Gets the plot element that corresponds to the specified point index.

Parameters

- **pointIndex**: **number**
  The index of the data point.

Inherited From
SeriesBase
Returns
any

hitTest

```
hitTest(pt: any, y?: number): HitTestInfo
```

Gets a **HitTestInfo** object with information about the specified point.

Parameters

- **pt**: **any**
  The point to investigate, in window coordinates.

- **y**: **number** OPTIONAL
  The Y coordinate of the point (if the first parameter is a number).

Inherited From
SeriesBase
Returns
HitTestInfo

initialize

```
initialize(options: any): void
```

Initializes the series by copying the properties from a given object.

Parameters

- **options**: **any**
  JavaScript object containing initialization data for the series.

Inherited From
SeriesBase
Returns
void

legendItemLength

```
legendItemLength(): number
```

Returns number of series items in the legend.

Inherited From
SeriesBase
Returns
number

measureLegendItem

```
measureLegendItem(engine: IRenderEngine, index: number): Size
```

Measures height and width of the legend item.

Parameters

- **engine**: **IRenderEngine**
  The rendering engine to use.

- **index**: **number**
  Index of legend item(for series with multiple legend items).

Inherited From
SeriesBase
Returns
Size

onRendered

```
onRendered(engine: IRenderEngine): void
```

Raises the **rendered** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.

Inherited From
SeriesBase
Returns
void

onRendering

```
onRendering(engine: IRenderEngine, index: number, count: number): boolean
```

Raises the **rendering** event.

Parameters

- **engine**: **IRenderEngine**
  The **IRenderEngine** object used to render the series.
- **index**: **number**
  The index of the series to render.
- **count**: **number**
  Total number of the series to render.

Inherited From
SeriesBase
Returns
boolean

pointToData

```
pointToData(pt: Point): Point
```

Converts a **Point** from control coordinates to series data coordinates.

Parameters

- **pt**: **Point**
  The point to convert, in control coordinates.

Inherited From
SeriesBase
Returns
Point

Events

rendered

Occurs when series is rendered.

Inherited From
`SeriesBase`
Arguments
`IRenderEngine`

rendering

Occurs when series is rendering.

Inherited From
`SeriesBase`
Arguments
`EventArgs`

WjFlexChartFibonacciTimeZones Class

File
`wijmo.angular2.js`
Module
wijmo/wijmo.angular2.chart.finance.analytics
Base Class
FibonacciTimeZones

Angular 2 component for the **FibonacciTimeZones** control.

The **wj-flex-chart-fibonacci-time-zones** component must be contained in a **WjFinancialChart** component.

Use the **wj-flex-chart-fibonacci-time-zones** component to add **FibonacciTimeZones** controls to your Angular 2 applications. For details about Angular 2 markup syntax, see Angular 2 Markup.

The **WjFlexChartFibonacciTimeZones** component is derived from the **FibonacciTimeZones** control and inherits all its properties, events and methods.

Constructor

- constructor

Properties

- altStyle
- asyncBindings
- axisX
- axisY
- binding
- bindingX
- chart
- collectionView
- cssClass
- endX
- hostElement
- initialized
- isInitialized
- itemsSource
- labelPosition
- legendElement
- levels
- name
- renderedNg
- renderingNg
- startX
- style
- symbolMarker
- symbolSize
- symbolStyle
- visibility
- wjProperty

Methods

- created
- dataToPoint
- drawLegendItem
- getDataRect
- getPlotElement
- hitTest
- initialize
- legendItemLength
- measureLegendItem
- onRendered
- onRendering
- pointToData

Events

- rendered
- rendering

Constructor

constructor

```
constructor(options?: any): FibonacciTimeZones
```

Initializes a new instance of the **FibonacciTimeZones** class.

Parameters

- **options**: **any** OPTIONAL
  A JavaScript object containing initialization data.

Inherited From
FibonacciTimeZones
Returns
FibonacciTimeZones

Properties

altStyle

Gets or sets the alternative style for the series. The values from this property will be used for negative values in Bar, Column, and Scatter charts; and for rising values in financial chart types like Candlestick, LineBreak, EquiVolume etc.

If no value is provided, the default styles will be used.

Inherited From
SeriesBase
Type
any

asyncBindings

Allows you to override the global **WjOptions.asyncBindings** setting for this specific component. See the **WjOptions.asyncBindings** property description for details.

Type
boolean

axisX

Gets or sets the x-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

axisY

Gets or sets the y-axis for the series.

Inherited From
`SeriesBase`
Type
`Axis`

binding

Gets or sets the name of the property that contains Y values for the series.

Inherited From
`SeriesBase`
Type
string

bindingX

Gets or sets the name of the property that contains X values for the series.

Inherited From
`SeriesBase`
Type
string

chart

Gets the **FlexChart** object that owns this series.

Inherited From
SeriesBase
Type
FlexChartCore

collectionView

Gets the **ICollectionView** object that contains the data for this series.

Inherited From
SeriesBase
Type
ICollectionView

cssClass

Gets or sets the series CSS class.

Inherited From
SeriesBase
Type
string

endX

Gets or sets the ending X data point for the time zones.

If not set, the ending X data point is calculated automatically. The value can be a number or a Date object (for time-based data).

Inherited From
FibonacciTimeZones
Type
any

hostElement

Gets the series host element.

Inherited From
`SeriesBase`
Type
SVGGElement

initialized

This event is triggered after the component has been initialized by Angular, that is all bound properties have been assigned and child components (if any) have been initialized.

Type
EventEmitter

isInitialized

Indicates whether the component has been initialized by Angular. Changes its value from false to true right before triggering the **initialized** event.

Type
boolean

itemsSource

Gets or sets the array or `ICollectionView` object that contains the series data.

Inherited From
`SeriesBase`
Type
any

labelPosition

Gets or sets the `LabelPosition` for levels in `FibonacciTimeZones` tool.

Inherited From
`FibonacciTimeZones`
Type
`LabelPosition`

legendElement

Gets the series element in the legend.

Inherited From
`SeriesBase`
Type
SVGGElement

levels

Gets or sets the array of levels for plotting.

Default value is [0, 1, 2, 3, 5, 8, 13, 21, 34].

Inherited From
`FibonacciTimeZones`
Type
number[]

name

Gets or sets the series name.

The series name is displayed in the chart legend. Any series without a name does not appear in the legend.

Inherited From
`SeriesBase`
Type
string

renderedNg

Angular (EventEmitter) version of the Wijmo **rendered** event for programmatic access. Use this event name if you want to subscribe to the Angular version of the event in code. In template bindings use the conventional **rendered** Wijmo event name.

Type
EventEmitter