Developer's Guide

Table of Contents

1 Getting Started

1.1 Overview

Welcome to Emprise JavaScript Charts. Constructed entirely in JavaScript the days of annoying plugin downloads and browser security warnings are gone. With genuine ease of use and complete customization Emprise JavaScript Charts provides you with the tools you need to publish your data quickly and in a variety of formats. With its wide range of interactive features, simple and straightforward implementation, and unparalleled functionality, Emprise JavaScript Charts is the clear first choice for all your charting needs.

Here's a quick sampling of just some of the features included:

- **Interactive:** Features such as Hints, Mouse Tracking, Mouse Events, Key Tracking and Events, Zooming, Scrolling, and Crosshairs raise interactivity and user experience in web charting to a new level.

- **Axis Scaling:** There's no need to determine your data range before hand. EJSChart will calculate and scale automatically to fit whatever data it is presented with.

- **Auto Zooming, Scrolling:** Too much data and not enough screen real estate? Show it all. Let your end users zoom in on the pieces they're most interested in. Axis locking for single axis zoom, scrolling and automatic axis scaling are all included.

- **Stackable Series:** Multiple chart series can be stacked and combined to fit many charting needs.

- **Multiple Chart Types:** Line, Area, Scatter, Pie, Bar and Function series are just the beginning. New series are just a few lines of JavaScript code away.

- **Ajax-Driven Data:** EJSChart supports XML-formatted data and loads data on the fly. New series can be added and data updated in real time without page reloads.

- **Compatible:** Built with compatibility in mind and tested on all major browsers, you can be assured your charts will function consistently for the broadest range of end users. See the full list of compatible browsers on our System Requirements page.

- **Plugin Free:** 100% Pure JavaScript Charting Solution. No more worries of incompatible plugin versions or confusing security warnings. EJSChart is pure JavaScript and requires no client installation.

- **Customizable:** Every aspect of the charting display can be configured and customized through well-documented properties and methods. Want to do more than just change the color of the background? Need a series type which doesn't already exist? EJSChart is fully customizable and extendable to provide the greatest flexibility and integration for existing site designs and needs.

1.2 Implementation Instructions

Ease of implementation was an important factor considered in the development of EJSChart. The process from purchase to user "wow" takes just a matter of minutes and can be completed following the few easy to follow steps detailed below. Designed for all users, no previous JavaScript knowledge is required to implement EJSChart or take advantage of its many features. The steps below will guide you through the installation and testing of the EJSChart library.

1. Create a new directory in the home directory of your webpage and name it EJSChart.
2. Copy the contents (all files and directories) of the `/dist/` folder provided in your EJSChart download package into your newly created EJSChart folder on your web server.
3. Open the webpage that you wish to place an example chart on. If you do not currently have one or wish to use a test page for this purpose one has been provided for you in the How To directory included with your download. This file is named mydemochart.html and can be edited in your preferred html editor or even notepad.
4. Copy `<script type="text/javascript"`

`src="/EJSChart/EJSChart.js"></script>` into the head section of your webpage. *This path can be modified as necessary to correctly point to EJSChart.js.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">

    <script type="text/javascript" src="/EJSChart/EJSChart.js"></script>

  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>
  </body>
</html>
```

5. Create a div on your page where you would like the chart to be displayed. The size of the chart can be specified within the div properties if desired. The div is created by pasting `<div id="myChart" style="width:450px; height:330px;"></div>` where desired in the body of your webpage. The style width and height properties can be adjusted as desired.

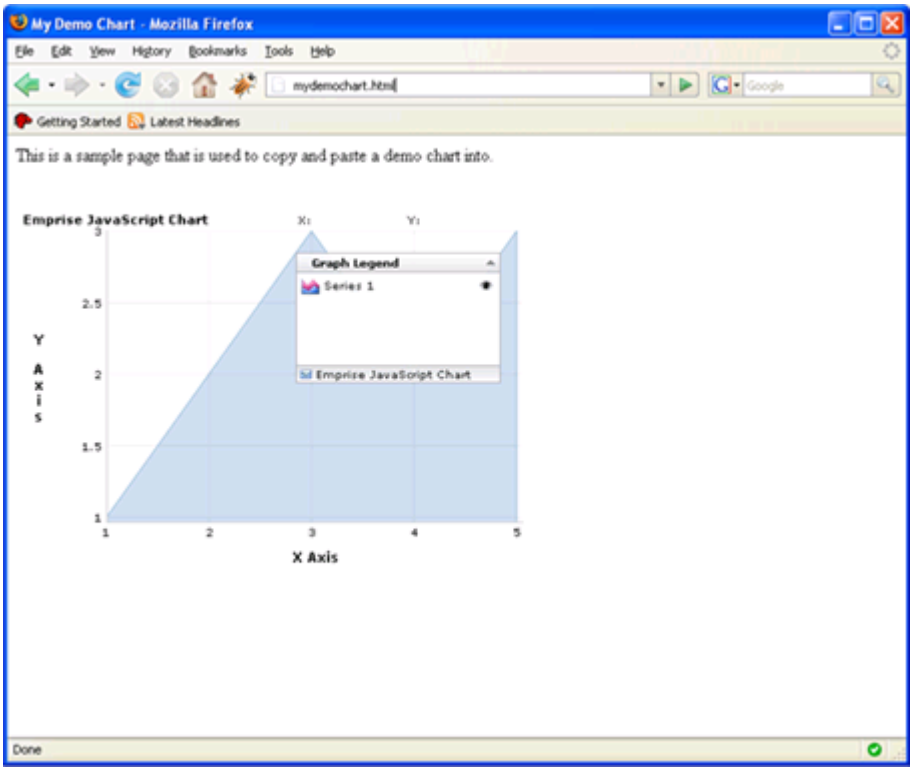
```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myChart" style="width:450px; height:330px;"></div>

  </body>
</html>
```

6. To turn the div into a chart paste the following sample code at the end of your webpage.

```
<script type="text/javascript">
  var chart = new EJSC.Chart("myChart");
      chart.addSeries(new EJSC.AreaSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,3]]))
      );
</script>
```

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myChart" style="width:450px; height:330px;"></div>

    <script type="text/javascript">
      var chart = new EJSC.Chart("myChart");
          chart.addSeries(new EJSC.AreaSeries(
              new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,3]]))
          );
    </script>
  </body>
</html>
```

7. Upload your edited webpage to your web server. Visit your webpage and view the demo chart located on the page where the div was placed.



1.3 Creating Your First Chart

Now that you have created and tested your first demonstration chart on your web server it is time to create your own chart. This demonstration will guide you through creating your first

chart and introduce you to some of the many available features and methods of customization available to you in Emprise JavaScript Charts. If you have not yet completed the Implementation Instructions it is recommended that you do so before continuing with this section as they guide you through uploading the necessary files to your web server.

1. Open the webpage that you wish to place your first custom chart on. If you do not currently have one or wish to use a test page for this purpose one has been provided for you in the How To directory included with your download. This file is named myfirstchart.html and can be edited in your preferred html editor or even notepad.
2. Copy `<script type="text/javascript" src="/EJSChart/EJSChart.js"></script>` into the head section of your webpage. *This path can be modified as necessary to correctly point to EJSChart.js.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts :: Customization
Example">

    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>

  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>
  </body>
</html>
```

3. Create a div on your page where you would like the chart to be displayed by pasting `<div id="myFirstChart" style=”width:600px; height:400px;"></div>` where desired in the body of your webpage.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts :: Customization
Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myFirstChart" style="width:600px; height:400px;"></div>

  </body>
</html>
```

4. The next step is to create the chart object. This is done at the end of your html file,

directly before the closing body tag; </body>. To begin define the text as JavaScript with `<script type="text/javascript">` and create a chart object on the next line using var `chart = new EJSC.Chart("myFirstChart");` Be sure that the div id as identified within the body of your webpage matches the title you place within the parenthesis when creating your chart. In this example it is 'myFirstChart'. Failure to create the appropriate div's for the chart to be placed in will result in errors and prevent your chart from being displayed.

The properties of your chart including its visual appearance and interactivity are highly customizable by adjusting its properties. Instructions on how to accomplish this can be found on the Customizing Your Chart page of this guide.

In this example code the chart has been customized to remove the legend and set a custom title by adding:

```
<script type="text/javascript">
  var chart = new EJSC.Chart(
    "myFirstChart",
    {
      show_legend: false,
      title: "My First Custom Chart"
    }
  );
```

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts :: Customization
Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myFirstChart" style="width:600px; height:400px;"></div>

    <script type="text/javascript">
      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

  </body>
</html>
```

5. Now that the chart has been created the series must be created. The series defines how

your data will be displayed in the chart you are creating. The types of series that are available may vary by license; more details on this can be found on the LICENSE.txt file.

When creating the chart you can also choose which format you will be providing the data in. This is accomplished with the use of different Data Handlers. The formats currently supported by EJSChart are XML file, JavaScript Array, CSV file, and CSV string data. Details on the implementation of each of these data handlers as well as sample code can be found on their respective pages in the Developer Guide help file.

The properties of your series, including its visual appearance and interactivity, are customized by adjusting its properties. Instructions on how to accomplish this can be found on the Customizing Your Chart page.

In this example code a bar chart was created using the Bar Series, the color was set to green, and the bar border width was set to five pixels. The data to be graphed was specified using the EJSC.ArrayDataHandler. To accomplish this the following code was added:

```
var myChartSeries =  new EJSC.BarSeries(
        new EJSC.ArrayDataHandler(
          [
            ["Month 1",1],
            ["Month 2",2],
            ["Month 3",3],
            ["Month 4",4],
            ["Month 5",5]
          ]
        )
      );
    myChartSeries.color = 'rgb(50,210,50)';
    myChartSeries.lineWidth = 5;
```

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts :: Customization
Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myFirstChart" style="width:600px; height:400px;"></div>

    <script type="text/javascript">

      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

      var myChartSeries =  new EJSC.BarSeries(
            new EJSC.ArrayDataHandler(
              [
                 ["Month 1",1],
                 ["Month 2",2],
                 ["Month 3",3],
                 ["Month 4",4],
                 ["Month 5",5]
              ]
            )
          );
        myChartSeries.color = 'rgb(50,210,50)';
        myChartSeries.lineWidth = 5;

  </body>
</html>
```

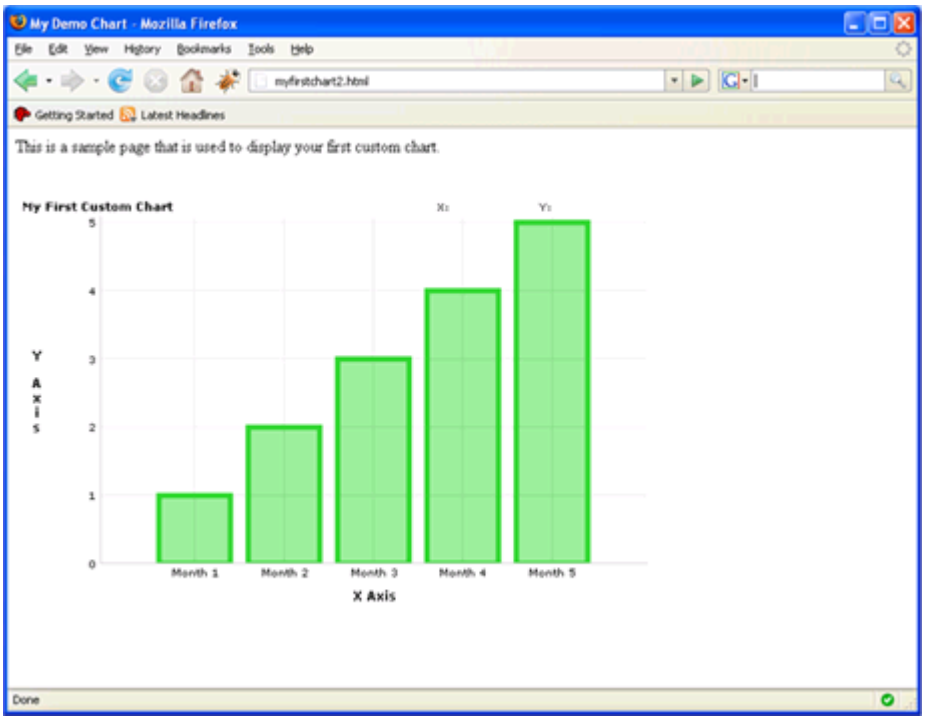
6. Once created the series must then be added to the chart using the addSeries method in the chart class.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts :: Customization
Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myFirstChart" style="width:600px; height:400px;"></div>

    <script type="text/javascript">

      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

      var myChartSeries =  new EJSC.BarSeries(
            new EJSC.ArrayDataHandler(
              [
                ["Month 1",1],
                ["Month 2",2],
                ["Month 3",3],
                ["Month 4",4],
                ["Month 5",5]
              ]
            )
          );
        myChartSeries.color = 'rgb(50,210,50)';
        myChartSeries.lineWidth = 5;

      chart.addSeries(myChartSeries);

  </body>
</html>
```

7. Close the JavaScript by inserting `</script>` at the end.

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="Emprise JavaScript Charts :: Customization
Example">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </script>
  </head>
  <body>
    <p>This is a sample page that is used to copy and paste a demo chart into.</p>

    <div id="myFirstChart" style="width:600px; height:400px;"></div>

    <script type="text/javascript">

      var chart = new EJSC.Chart(
        "myFirstChart",
        {
          show_legend: false,
          title: "My First Custom Chart"
        }
      );

      var myChartSeries =  new EJSC.BarSeries(
            new EJSC.ArrayDataHandler(
              [
                ["Month 1",1],
                ["Month 2",2],
                ["Month 3",3],
                ["Month 4",4],
                ["Month 5",5]
              ]
            )
          );
        myChartSeries.color = 'rgb(50,210,50)';
        myChartSeries.lineWidth = 5;

      chart.addSeries(myChartSeries);

    </script>

  </body>
</html>
```

8. That's it! Upload the webpage to your web server and you're done.



1.4 Customizing Your Chart

With Emprise JavaScript Charts, the customization options of your chart are endless. This includes visual appearance, which can be modified to integrate fully with any theme or design, as well as chart interactivity, which can range from including user capabilities such as auto zooming and custom hint captions to hidden-axis view only chart displays. Your chart can be customized on the chart and series level via the modification of the chart and series properties.

The first customization options available to you are at the chart level. They can be specified when creating the chart or alternatively after the chart object has been established. The properties available for editing and their syntax can be found in the Chart Properties section of the help documentation. In addition, examples of their implementation with sample code and the resulting effect on chart display or interactivity are available on the in the /examples/ directory of the distribution package. There are also additional examples available online at http://www.ejschart.com/examples/

The following are a few quick examples of modifying commonly used properties both at and after chart creation:

- Modification of the x and y axis labels during chart creation:

```
var chart = new EJSC.Chart("myChart", {x_axis_caption: "Month",
                                       y_axis_caption:
"Temperature"});
```

- Modification of the x and y axis labels after chart creation:

```
var chart = new EJSC.Chart("myChart");
    chart.setXAxisCaption("Month");
    chart.setYAxisCaption("Temperature");
```

The properties of each series on a chart may also be customized. This is accomplished with the editing of properties in the same way as was done to customize chart properties; either at the time of series creation or following creation. The properties available for editing and their syntax can be found on the Properties page of each of the different series help sections. In addition, examples of their implementation with sample code and the resulting effect on chart display or interactivity are available on the in the /examples/ directory of the distribution package. There are also additional examples available online at http://www.ejschart.com/examples/

The following are a few quick examples of modifying commonly used properties both at and after chart creation:

- Modification of the series lineWidth property during creation:

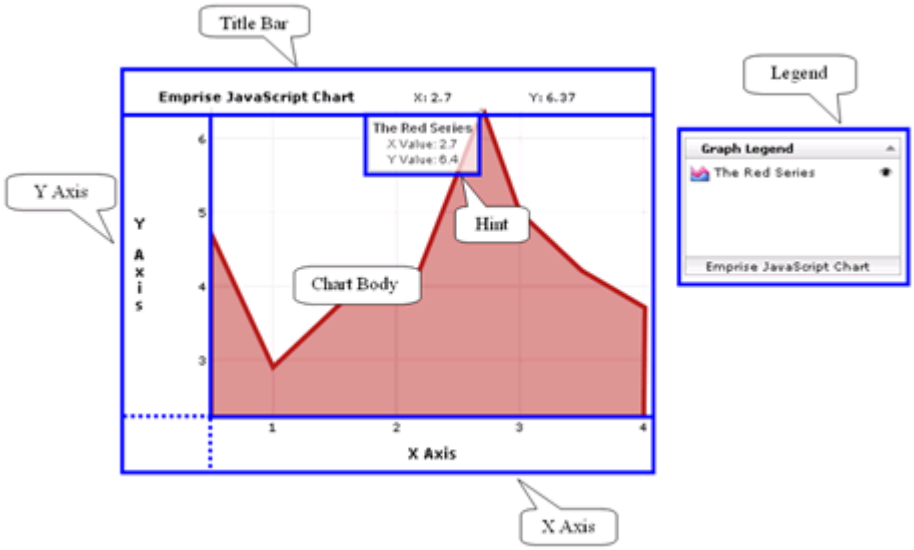
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin, {lineWidth:
4});
```

- Modification of the lineWidth property after creation:

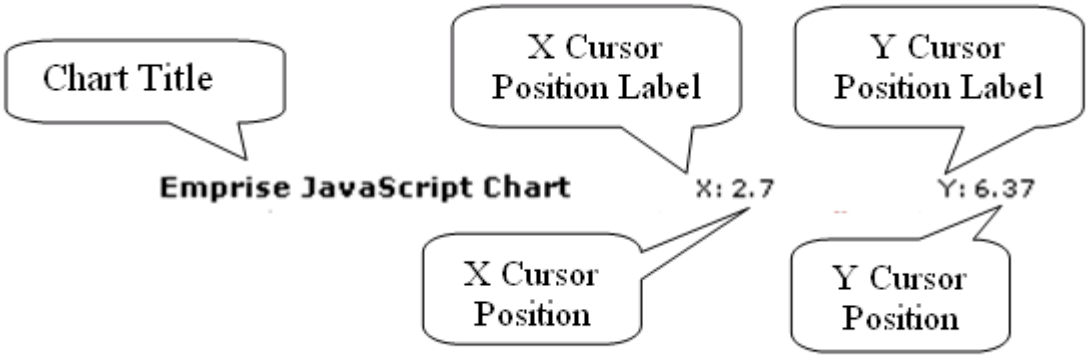
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin);
    myChartSeries.setLineWidth(4);
```

In order to fully understand the power of the properties available for customizing it is important to become familiar with the various components of the chart. The following diagram identify many of these components:

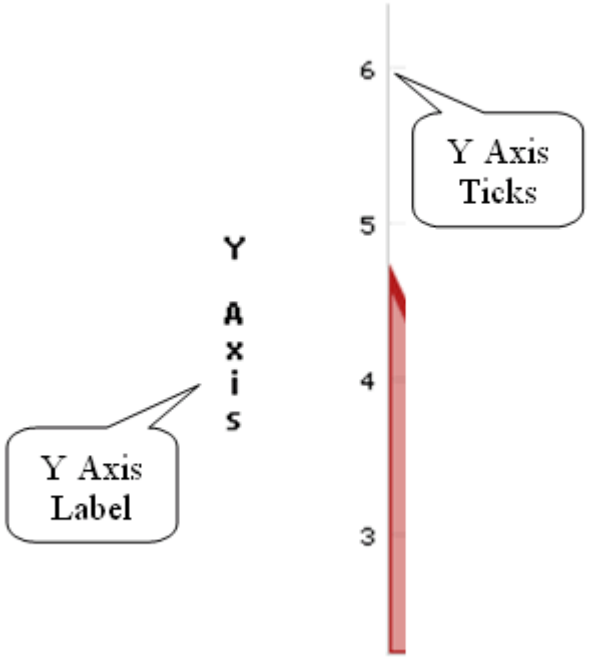
Anatomy of a Chart



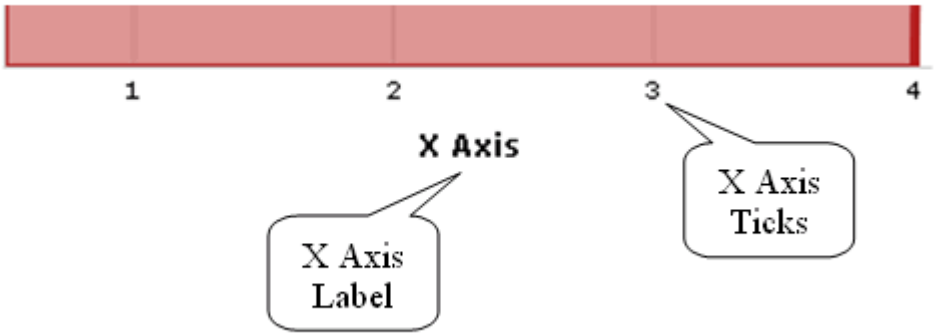
1.4.1 Titlebar



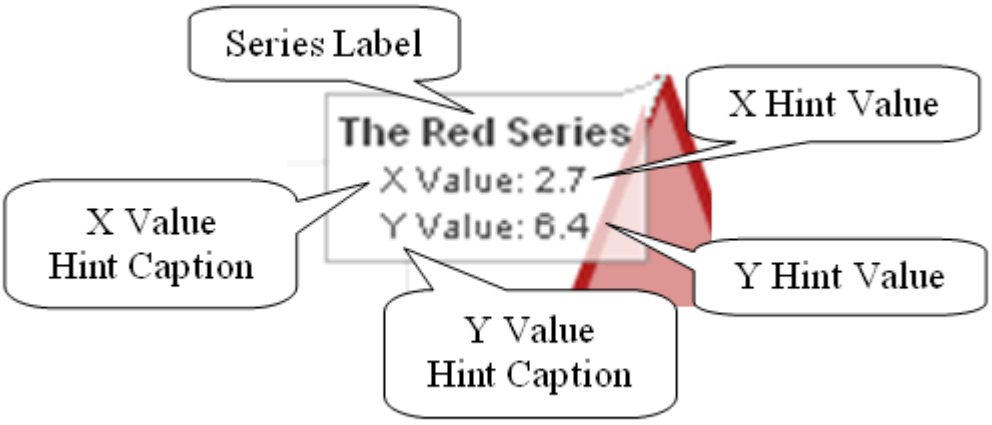
1.4.2 Y Axis



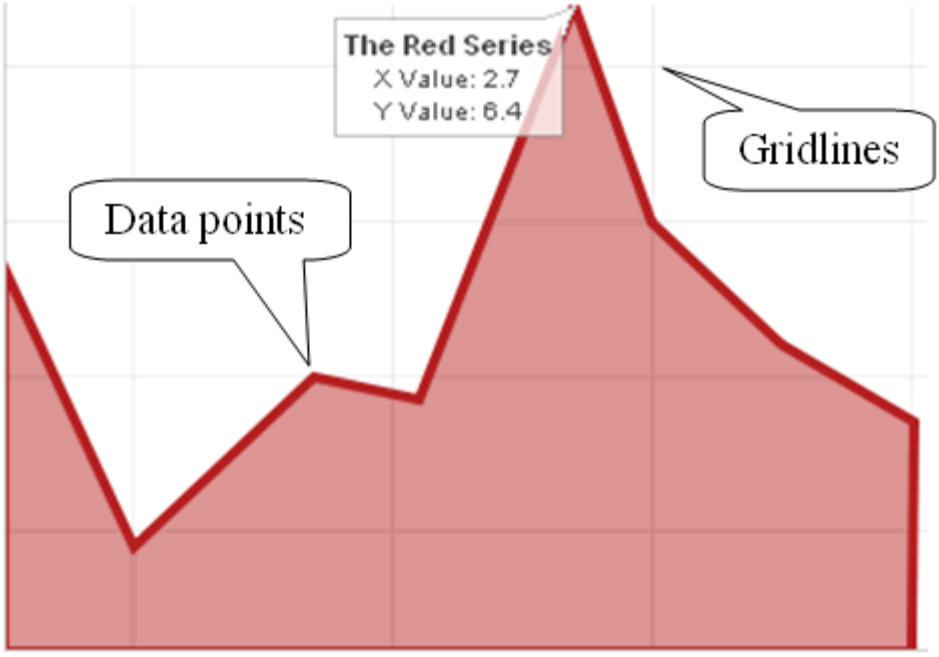
1.4.3 X Axis



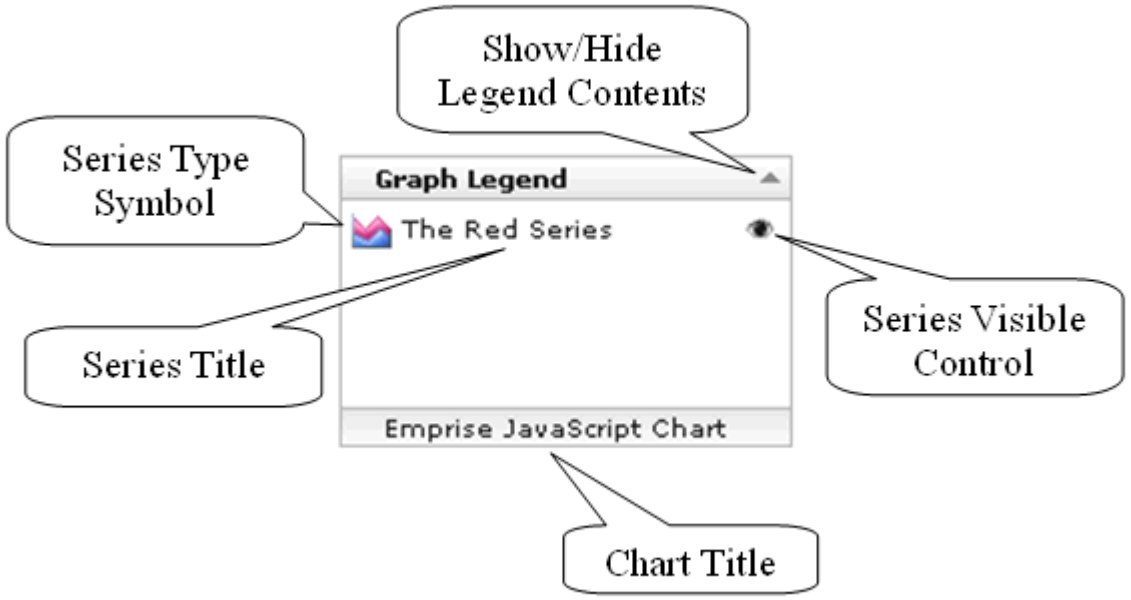
1.4.4 Hint



1.4.5 Chart Body



1.4.6 Legend



2 Deployment

The Emprise JavaScript Charts distribution package includes a compressed and obfuscated version of the source code for easy deployment. The entire /dist/ directory is meant to be placed on a web site, as is, with no modification necessary.

If you have access to the source code (Developer and Enterprise editions) and have made modifications, there is a utility web page located in the /packer/ directory which hosts a compression script powered by Dean Edward's Packer (original source: http://dean.edwards.name/packer/)

This utility must be used to compress and obfuscate the JavaScript source code before making it live or distributing it within your own application.

3 Changes

3.1 Changes in version 1.3

Additions

- Added AreaSeries.closeLine property which determines if the line should return to the zero plane and draw as a closed shape.
- Added PieSeries.findCenterOfCurve(point) method
- Added support for SVG exporting

Modifications

- Modified axis to write out ticks and labels less often
- Line and scatter series are now split into smaller draws in order to support larger data sets in IE
- BarSeries point selection now accounts for proximity_snap setting and the bar's line width
- Added support for text labels (bins) to TrendSeries

Bug Fixes

- Updated color conversion routine to fix issues in IE
- Added additional logic to ajax data handlers to prevent double loading
- Corrected resize issue in IE which could cause an 'Invalid Argument' error
- Updated extremes calculations to correct issues with negative numbers
- getYExtremes() now correctly returns the y axis extremes
- Added additional offset checks in point location routines to account for quirks mode in various browsers
- Fixed issues with xml data handler and processing string values
- PieSeries data points now correctly save and display their labels if defined
- Corrected drawing issues in BarSeries with negative numbers
- Fixed BarSeries drawing in IE with a large number of points
- Corrected BarSeries draw in IE to close the bar line
- Updated PieSeries to reset its available colors correctly when reloading

3.2 Changes in version 1.2.1

Additions

- Added selectPoint and clearSelectedPoint methods to the chart.

Modifications

- Enhanced support for chart with large numbers of series.
- Enabled initial implementation of Series.delayLoad (default true) which controls when a series triggers its data handler to begin data retrieval
- Updated PiePoint and PieSeries to correctly render float values for pie pieces
- Updated PieSeries to handle more 100% piece drawing cases

Bug Fixes

- Fixed issues with series reloading where auto calculated extremes were not reset with new data (affected all series)
- Fixed print rendering issues in all browsers except Opera. The chart now renders as show on screen when printing.
- Fixed XYPoint class to properly recognize the Y value as a number as opposed to a string.
- Fixed inheritance issue with ArrayDataHandler
- Fixed issues with clearing the chart when all loaded series are set invisible
- Fixed PieSeries reload method to reset available colors
- Fixed BarSeries treeLegend property to function properly when useColorArray is false
- Fixed BarSeries reload method to reset available colors
- Fixed range drawing in AnalogGaugeSeries to correctly connect arcs before filling
- Fixed issue with x axis labels not aligning correctly when allow_interactivity was false

3.3 Changes in version 1.2

Additions

- Added classes to better control and track axis bins (text labels)
- Added support for y axis bins
- Added support for shared bins between series (unused bins)
- Added coordinate property to Chart.y_zero_plane and Chart.x_zero_plane to allow zero plane to fall somewhere other than 0
- Added Chart.force_static_points_x and Chart.force_static_points_y (force_static_points works as before, for x only) to force bins when data is numeric
- Implemented Chart.y_axis_extremes_ticks
- Added support for drawing minor tick marks on the Y axis and Y axis
- Added Chart.onXAxisNeedsTicks and onYAxisNeedsTicks to allow for custom tick configurations
- Added support for auto sort in all applicable series / data handlers. This is now set by default and may be turned off if data is correctly sorted prior to being handed to the data handler.
- Added support for color specifications as rgb(0,0,0), rgba(0,0,0,100) and hex (#000000) everywhere colors may be specified
- Added support for mouse wheel to zoom (controlled by allow_interactivity, allow_zoom, allow_mouse_wheel_zoom)
- Added Chart.addYAxisBin, Chart.addXAxisBin, Chart.removeYAxisBin, Chart.removeXAxisBin methods to support manual definition (and order) of axis bins
- Added EJSC.XMLStringDataHandler class
- Added drawPoints, pointSize, pointColor, pointBorderSize, pointBorderColor properties to EJSC.LineSeries and EJSC.AreaSeries
- Added support for new zero plane coordinate property to AreaSeries and BarSeries
- Added tree-style legend for PieSeries
- Added tree-style legend option for BarSeries (default true when useColorArray is true and treeLegend is left undefined)
- Added position and height/width properties to PieSeries to support multiple pie series per chart
- Added findCenter method to PieSeries to find the center point of a given piece
- Added getPoints method to PieSeries (to allow findCenter calls to provide a valid piece object)
- Added orientation property to BarSeries and support for horizontal bar charts
- Added total_value property and getTotalValue, setTotalValue, resetTotalValue methods to PieSeries
- Added [total] and [percent] as hint text replacement options for PieSeries
- Added x_cursor_position_formatter and y_cursor_position_formatter properties to chart to allow for a different formatter for mouse position display

Modifications

- Consolidated and cleaned up xml parsing routines
- Consolidated and cleaned up csv parsing routines
- Cached many often used math methods for performance
- Modified extremes to account for manually added bins with no associated data in series
- Rewrote data handlers to better support text labels (bins)
- Modified pie series to correctly use lineOpacity
- Updated pie piece drawing to one path creation per piece
- Added check in AnalogGaugeSeries to avoid resetting innerHTML unless the label has actually changed.
- Added check of show_hints before triggering onShowHint event

Bug Fixes

- Added media attribute to link tag to fix printing issues in firefox/netscape
- Added correct headers when sending an xml request as POST
- Fixed case when last tick is missing while using x_axis_tick_count
- Fixed spelling - getZoomBoxCoordinates
- Updated lineOpacity to 100 (from 1) in PieSeries so lines draw correctly
- Added check for single 100% piece to remove line when drawn (fixed full circle drawing)
- Reversed __intercept/__slope properties in TrendSeries so they represent the correct values
- Updated number formatter to check for -0 as final output and remove - sign
- onDblClickPoint and onAfterSelectPoint both now function correctly when show_hints is false
- Fixed unterminated string error in TrendSeries
- Updated style to remove border when axes are not visible (ie, pie or gauge only charts)
- Fixed constant resize issue with pie charts (and anything with x axis hidden) in IE6/opera
- Fixed issue with series drawing multiple times when added and redraw is false

3.4 Changes in version 1.1

Additions

- Added new series type AnalogGaugeSeries
- Added onBeforeBeginZoom, onBeforeEndZoom and onAfterShowCrosshairs events to Chart
- Added support for local loading of chart code in IE 7 and IE6 with certain versions of the XMLHttpRequest ActiveX control.
- Added new axis styling properties to Chart: x_axis_size, x_axis_className, x_axis_tick_className, x_axis_stagger_ticks, x_axis_tick_count, y_axis_size, y_axis_className, y_axis_tick_className, y_axis_tick_count.
- Added Chart.setZoom and getZoom methods to in order to get the current zoom and set zoom to a particular range
- Added Chart.getMinMaxYInXRange method to programatically find the min and max Y values in a given X range
- Added Chart.findClosestPointInSeries method to programatically find the closest point to the coordinates (x,y) in a particular series
- Added Chart.getZoomBoxCoordinates method to get the current zoom box coordinates
- Added Chart.displayZoomBox method to programatically display the zoom box
- Added BarPoint class to support additional BarSeries functionality
- Added the ability to draw bars with different colors based on ranges
- Added the ability to adjust the spacing between bars using the new intervalOffset property.
- Added the ability to switch between grouped and overlayed bars with BarSeries
- Added a color pool to BarSeries for individually colored bars (similar to PieSeries)
- Added onBarNeedsColor to BarSeries for more advanced bar coloring options
- Added setDefaultColors, addRange, deleteRange, clearRanges, setIntervalOffset and setGroupedBars methods to BarSeries
- Added support for 'exponential' and 'logarithmic' types in TrendSeries
- Added showGrid/hideGrid methods with an optional redraw parameter to prevent immediate redrawing
- Optimized trend series calculation routines
- Blank X or Y axis captions now trigger the axis to be resized giving the chart additional space
- Added the ability to specify only changed members of child objects via the options property
- Added Chart.convertPointToPixel method to calculate document pixel top/left of a point in chart units
- Added Chart.convertPixelToPoint method to calculate chart point from a document pixels

- Added base AjaxDataHandler class, XMLDataHandler and CSVFileData handler now descend from this class.
- Added requestType property and setRequestType method to AjaxDataHandler for changing between GET and POST.
- Added urlData property and setUrlData method to AjaxDataHandler to allow assignment of POST data and GET url parameters
- Added onNeedsData event and setXMLData method to AjaxDataHandler to support 3rd party Ajax libraries for data retrieval
- Added reload flag to setUrl method (default = false) in AjaxDataHandler to allow for immediate data retrieval / series reload.
- Added Chart.allow_hide_error property (default = false) which allows error messages to be hidden and overwritten with non-error messages
- Added additional error checking and reporting into xmlrequestpool, errors occuring during series data retrieval via XML or CSVFile data handlers will be displayed on the chart.
- Added EJSC.utility.XMLRequestPool.fatalErrors property to define which HTTP codes cause a XMLHttpRequest to fail.
- Added redraw flag to Chart.addSeries in order to postpone full redraw until all series have been added.
- Added Chart.remove() method which completely deletes a chart from the page.

Modifications

- Moved a number of private properties into the EJSC namespace for easier patching.
- Modified load/unload events to use utility.attachEvent method so that the methods are detached automatically
- Changed non-IE browser load of stylesheet to insert link tag instead of force load stylesheet via ajax
- Reduced minimum height to 60 px
- Tick container now positioned absolute in order to support additional styling options.
- Added chart parameter to onShowCrosshairs event
- Removed case sensitivity requirement for locating support files.

Bug Fixes

- Consolidate references to html and head tags, clear references on unload to fix IE leak
- Fixed issue with attachEvent in Opera causing errors
- Added cleanup routine to series in order to remove references to legend items and fix IE leaks
- Fixed canvas cover positioning
- x_axis container left coordinate is now set to fix issues with table/float layouts causing odd shifts in IE
- doRecalcExtremes now called in resize method in order to re-adjust padding appropriately to the new chart size
- Fixed issues with drawing first series added when extreme values are static
- Fixed issues in recalculation of extreme values when no series data is available
- Fixed offset issues with hints and point selection
- Fixed issues with point selection routine when no point is selected
- Fixed onBeforeDblClick event to send correct reference to chart
- Fixed issue with array data handler loading when array is empty
- Fixed scatter series to not draw line through center of circle
- Fixed and optimized several issues with drawing and point selection within BarSeries with many bars
- Fixed issues with calculating spacing and widths with multiple bar series in a single chart
- Fixed issues with drawing and point selection in bar series when bar is only partially in view

- Fixed style issues with legend header and legend captions in IE

3.5 Changes in version 1.0.1

Additions

- Added series.opacity property
- Added series.setOpacity method
- Added series.lineOpacity property
- Added series.setLineOpacity property
- Added useUTC property to DateFormatter (default value == true)
- Added timezoneOffset property to DateFormatter (default value == undefined) Only applicable when useUTC is true
- Added series.legendIsVisible property (default value == true)
- Added series.showLegend and series.hideLegend methods
- Added chart.legendTitle property and chart.setLegendTitle method

Modifications

- Modified to remove all code inside target div
- Attach events to chart container to prevent selection
- Moved lineWidth property and setLineWidth method into base Series class
- Modified LineSeries to use lineOpacity properties
- Modified AreaSeries to use opacity and lineOpacity properties
- Modified ScatterSeries to use lineWidth, opacity and lineOpacity properties
- Modified PieSeries to use lineWidth, opacity and lineOpacity properties
- Modified BarSeries to use opacity and lineOpacity properties
- Modified FunctionSeries and TrendSeries to use lineOpacity property
- Text selection is now cancelled when drag/selection originates in the chart (except Opera)

Bug Fixes

- Fixed hint and point selection issues when the chart is inside a scrollable container
- Added check and alert if attachEvent fails
- Setting y_min/max, x_min/max during chart creation now correctly assign extremes
- Fixed scaling errors when only a single Y or X value has been specified
- Added call to onBeforeZoom when double clicking to allow canceling of zoom out
- Added call to onAfterZoom when double clicking to zoom out
- Fixed BarSeries line drawing on Internet Explorer
- Fixed automatic resizing of y axis caption in IE when using setYAxisCaption()
- Fixed style sheets to completely hide axis when turned off
- Fixed style sheets to hide key grabber element and remove border and background from cursor position elements.
- Fixed issues with long series titles wrapping legend items
- Added additional exception handling to unload methods and storage and clearing of auto-resize interval
- Fixed errors when attempting to load empty data sets (applicable to all data handlers)
- Fixed error when loading BarSeries with a single point

3.6 Changes in version 1.0

Additions

- Added an optional redraw flag into Chart.removeSeries in order to allow for multiple removes between chart redraws.
- Added support for user-defined data associated with a point
- Added support for reading user-defined data from full and short xml formats
- Added the setTitle method to the base Series class.
- Added getDataHandler method to base Series class.
- Added getUrl and setUrl methods to XMLDataHandler and CSVFileDataHandler classes.
- Added getArray and setArray methods to ArrayDataHandler class.
- Added getCSV and setCSV methods to CSVStringDataHandler class.
- Added coloredLegend property to Series to specify whether the series legend text should inherit the series color
- Added setColoredLegend method to Series in order to update the coloredLegend property after series creation

Modifications

- Updated chart resize methods to monitor the chart container and update whenever necessary, this adds greater compatibility with other 3rd party packages such as Dojo when the chart is placed inside a window class.
- Chart.removeSeries() now deletes the series object which was removed.
- Modified series legend items to have their text color match the series color.
- Modified series legend captions to not wrap and display full text with a hint.

Bug Fixes

- Fixed event attachments to global objects such as document.onmouseup to use attachEvent / addEventListener
- Chart.removeSeries() now correctly removes the legend objects without error.
- Fixed an issue where tall charts could lose interactivity.

4 Data Formats

The following is a brief summary of the data formats supported by Emprise JavaScript Charts. For additional information please see the individual class help files and example code available at http://www.ejschart.com/help/

XML

The EJSC.XMLDataHandler class will load an XML file containing chart data using AJAX. The following XML formats are supported:

Full:

```
<graph>
  <plot>
    <point x="" y="" />
  </plot>
</graph>
```

Short:

```
<G>
  <L>
    <P x="" y="" />
  </L>
</G>
```

Compact:

```
<G>
  <L values="X|Y,X|Y,X|Y" />
</G>
```

Array

The EJSC.ArrayDataHandler class will load chart data from a JavaScript array. The basic format of the array is as follows:

```
[
 [X Value, Y Value],
 [X Value, Y Value]
]
```

CSV (Comma Separated Values)

The EJSC.CSVFileDataHandler and EJSC.CSVStringDataHandler classes support a comma separated list of data points. EJSC.CSVFileDataHandler will load the data points list from a file on the server using AJAX. EJSC.CSVStringDataHandler takes a string in its constructor specifying the CSV text. The data is supported as x y value pairs (where x and y are separated by a pipe | symbol) as shown below:

```
X|Y,X|Y,X|Y
```

4.1 XML - Full

The full xml format is the most verbose and the most human readable of the supported formats. This format is ideal for experimenting with chart configuration and charts with smaller datasets.

General Usage:

```
<graph>
   <plot>
      <point x="1" y="1" />
      <point x="2" y="2" />
      <point x="3" y="3" />
      <point x="4" y="4" />
   </plot>
</graph>
```

Pie Series Data:

The above example is used for the majority of currently supported data series types. The exception to this is the EJSC.PieSeries which requires only an "x" attribute and adds an additional "label" to each point tag (the y attribute is not applicable to EJSC.PieSeries and will be ignored if specified):

```
<graph>
   <plot>
      <point x="100" userdata="" label="blue widgets" />
      <point x="200" userdata="" label="red widgets" />
      <point x="50" userdata="" label="green widgets" />
   </plot>
</graph>
```

Text Labels:

At times, numeric labels and the auto scaling options in the chart are not necessary, not available or not desired for display data. A EJSC.BarSeries for instance, which is used to compare number of widgets sold by color. The chart supports providing text as the X value, rather than a number:

```
<graph>
   <plot>
      <point x="Blue" y="100" />
      <point x="Red" y="200" />
      <point x="Green" y="50" />
   </plot>
</graph>
```

User-Defined Data:

When displaying custom hints and implementing drill down style reports it is often necessary to have an additional piece of data associated with each data point such as a database id value. The full and short xml formats support the **userdata** property which is read in and assigned to the associated EJSC.Point descendant. The value specified is entirely up to the developer. It could be a set of integers representing a database primary key value, a url which points to drill-down data or even a javascript function. The **userdata** property of a point is available during point-related events in the chart such as EJSC.Chart.onDblClickPoint, EJSC.Chart.onAfterSelectPoint, etc.

```
<graph>
   <plot>
      <point x="1" y="100" userdata="WHERE PointId=10432" />
      <point x="2" y="200" userdata="./drillDown2.xml" />
      <point x="3" y="50" userdata="alert('this is the userdata');" />
   </plot>
</graph>
```

4.2 XML - Short

The short XML format provides the same functionality as the full format but with less transfer overhead (there is no reduction in the processing overhead of extracting the point data, see the Array and CSV formats for more concise options).

The general format is as follows:

```
<G>
   <L>
      <P x="" y="" userdata="" />
   </L>
</G>
```

The <G> tag relates to the <graph> tag in the full format, the <L> tag relates to the <plot> tag and <P> is equivalent to <point>.

Support for pie charts is provided in the same manner as with the full format, i.e.

```
   <P x="" label="" />
```

4.3 XML - Compact

The compact xml format is geared towards larger data sets which need to reduce the overhead associated with transfer and extraction of chart data via a more verbose xml format.

The data is provided in csv-like format and stored in the "values" attribute of the <L> tag:

```
<G>
   <L values="1|1,2|2,3|3" />
</G>
```

For the majority of series, the above example would be used to provide X,Y coordinates.

5 API Reference

5.1 EJSC

Top level namespace for all classes and variables used by the Emprise JavaScript Charts package. Use of this namespace prevents variable name collisions with other available JavaScript packages.

5.1.1 Properties

5.1.1.1 DefaultImagePath

Definition

string **DefaultImagePath** = "images/"

Description

Relative path defining the path to all images used by EJSC classes.

5.1.1.2 DefaultColors

Definition

array **defaultColors** = [
"rgb(120,90,59)",
"rgb(53,115,53)",
"rgb(178,87,56)",
"rgb(203,143,71)",
"rgb(55,106,155)",
"rgb(205,197,51)",
"rgb(209,130,139)",
"rgb(159,153,57)",
"rgb(206,173,136)",
"rgb(191,132,72)",
"rgb(151,135,169)",
"rgb(140,48,51)",
"rgb(59,144,187)",
"rgb(197,190,104)",
"rgb(109,136,79)",
"rgb(144,100,144)",
"rgb(181,94,94)",
"rgb(59,144,144)",
"rgb(204,136,92)",
"rgb(139,167,55)",
"rgb(205,171,66)",
"rgb(150,184,211)"
]

Description

This array is used in each chart to specify available series colors. Add to or modify this array in order to define a single set of default colors used by all charts on a page.

5.1.1.3 DefaultBarColors

Definition

```
array defaultBarColors = [
        "rgb(120,90,59)",
        "rgb(53,115,53)",
        "rgb(178,87,56)",
        "rgb(203,143,71)",
        "rgb(55,106,155)",
        "rgb(205,197,51)",
        "rgb(209,130,139)",
        "rgb(159,153,57)",
        "rgb(206,173,136)",
        "rgb(191,132,72)",
        "rgb(151,135,169)",
        "rgb(140,48,51)",
        "rgb(59,144,187)",
        "rgb(197,190,104)",
        "rgb(109,136,79)",
        "rgb(144,100,144)",
        "rgb(181,94,94)",
        "rgb(59,144,144)",
        "rgb(204,136,92)",
        "rgb(139,167,55)",
        "rgb(205,171,66)",
        "rgb(150,184,211)"
]
```

Description

Defines the pool of default colors available for bar series bars.

5.1.1.4 DefaultPieColors

Definition

```
array defaultPieColors = [
        "rgb(120,90,59)",
        "rgb(53,115,53)",
        "rgb(178,87,56)",
        "rgb(203,143,71)",
        "rgb(55,106,155)",
        "rgb(205,197,51)",
        "rgb(209,130,139)",
        "rgb(159,153,57)",
        "rgb(206,173,136)",
        "rgb(191,132,72)",
        "rgb(151,135,169)",
        "rgb(140,48,51)",
        "rgb(59,144,187)",
        "rgb(197,190,104)",
        "rgb(109,136,79)",
        "rgb(144,100,144)",
```

```
        "rgb(181,94,94)",
        "rgb(59,144,144)",
        "rgb(204,136,92)",
        "rgb(139,167,55)",
        "rgb(205,171,66)",
        "rgb(150,184,211)"
]
```

Description

Defines the pool of default colors available for pie pieces.

5.2 EJSC.Chart

Chart class that holds all the generic information for each chart that is being created.

The constructor expects the id of a DOM object (preferrably a div) and an optional set of object properties. **NOTE:** The contents of the DOM object will be cleared before rendering the chart.

Constructor

```
EJSC.Chart( string id [, object options] )
```

Example

var myChart = **new** EJSC.Chart("chart", {title:"My Chart"});

Defining Chart Properties and Events

Chart properties may be set individually after the chart has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var myChart = new EJSC.Chart("chartDIV");
myChart.allow_zoom = false;
myChart.show_legend = false;
myChart.onDblClickPoint = function(point) {
        alert("Point Clicked: " + point.x + "," + point.y);
}
```

Setting properties in batch

```
var myChart = new EJSC.Chart(
        "chartDIV",
        {
                allow_zoom: false,
                show_legend: false,
                onDblClickPoint: function(point) {
                        alert("Point Clicked: " + point.x + "," + point.y);
                }
        }
);
```

5.2.1 Properties

5.2.1.1 allow_interactivity

Definition

boolean **allow_interactivity** = true

Description

Defines if the chart can be interacted with (ie zooming, moving, point selection, etc.).

Example

>> Turn off all interactivity for the chart

```
var chart = new EJSC.Chart(
                "chart",
                {allow_interactivity: false}
        );
```

5.2.1.2 allow_mouse_wheel_zoom

Definition

boolean **allow_mouse_wheel_zoom** = true

Description

Defines if the chart can be zoomed via the scroll wheel on the mouse. This is automatically disabled if allow_interactivity or allow_zoom is set to false.

Example

>> Turn off wheel zooming for the chart.

```
var chart = new EJSC.Chart(
                "chart",
                {allow_mouse_wheel_zoom: false}
        );
```

5.2.1.3 allow_zoom

Definition

boolean **allow_zoom** = true

Description

Defines if the chart can be zoomed. This is automatically disabled if allow_interactivity is set to false.

Example

>> Turn off zooming for the chart.

```
var chart = new EJSC.Chart(
                "chart",
                {allow_zoom: false}
        );
```

5.2.1.4 allow_hide_error

Definition

boolean **allow_hide_error** = false

Description

Defines whether the chart can overwrite an error message with a non-error message or hide the error message after a brief period of time.

Example

>> Allow all messages, including errors, to be hidden after a brief period of time

```
var chart = new EJSC.Chart(
                "chart",
                { allow_hide_error: true }
        );
```

5.2.1.5 auto_find_point_by_x

Definition

boolean **auto_find_point_by_x** = false

Description

Defines if the select point routine should select points based only on the X position of the cursor.

Example

>> Allow point selection to occur on click anywhere within the Y axis, find the closest point based solely on X.

```
var chart = new EJSC.Chart(
                "chart",
                {auto_find_point_by_x: true}
        );
```

5.2.1.6 auto_zoom

Definition

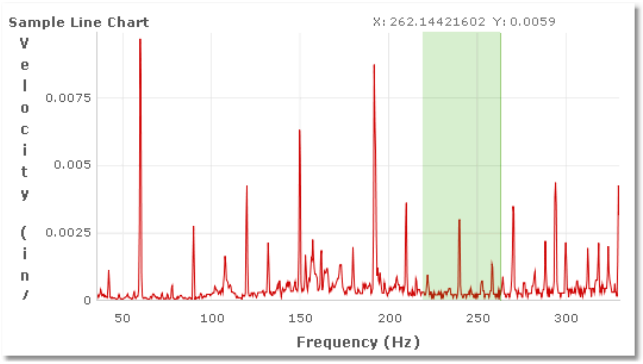
string **auto_zoom** = undefined

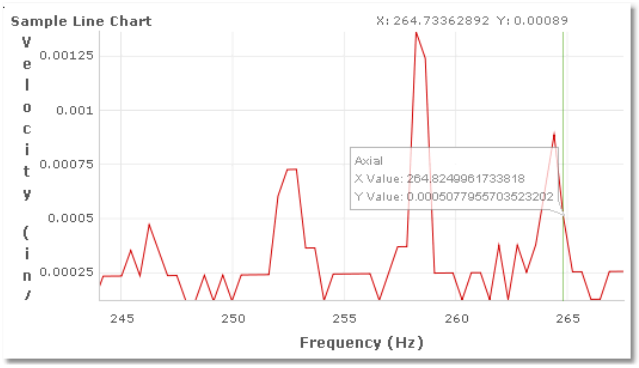
Valid property values:
'x' - **NOT IMPLEMENTED**
'y'

Description

Defines if the chart should auto-select the zoom area based on the X or Y coordinates selected by the user. The other coordinates are automatically selected based on best fit for the data defined in the range selected.

Example





>> Allow user to select beginning and ending X values to zoom, then auto-scale the Y axis according to the data.

```
var chart = new EJSC.Chart(
        "chart",
        { auto_zoom: "y" }
);
```

5.2.1.7 force_static_points

DEPRECIATED See force_static_points_x and force_static_points_y

Definition

boolean **force_static_points** = false

Description

Defines if the chart should force ticks to match up to every point by converting X data to strings.

Example

>> Display every X axis data point, essentially disable auto axis scaling.

```
var chart = new EJSC.Chart(
                "chart",
                {force_static_points: true}
        );
```

5.2.1.8 force_static_points_x

Definition

boolean **force_static_points_x** = false

Description

Defines if the chart should force ticks to match up to every point by converting X data to strings.

Example

>> Display every X axis data point, essentially disable auto axis scaling.

```
var chart = new EJSC.Chart(
                "chart",
                {force_static_points_x: true}
        );
```

5.2.1.9 force_static_points_y

Definition

boolean **force_static_points_y** = false

Description

Defines if the chart should force ticks to match up to every point by converting Y data to strings.

Example

>> Display every Y axis data point, essentially disable auto axis scaling.

```
var chart = new EJSC.Chart(
                "chart",
                {force_static_points_y: true}
        );
```

5.2.1.10 legendTitle

Definition

string **legendTitle** = "Chart Legend"

Description

Defines the caption displayed in the chart legend title bar. To modify this caption after chart creation, use Chart.setLegendTitle()

5.2.1.11 proximity_snap

Definition

integer **proximity_snap** = 5

Description

Determines the maximum number of pixels away from a point the cursor can be for point selection and hints.

Example

>> Allow clicks to trigger point selection up to 9 pixels away from the actual point.

```
var chart = new EJSC.Chart(
                "chart",
                {proximity_snap: 8}
        );
```

5.2.1.12 show_crosshairs

Definition

object **show_crosshairs** = {
    x: true,
    y: true
}

Description

Defines if crosshairs should be shown on the chart at the current mouse coordinates. May be disabled for X and Y independantly. This is automatically disabled if allow_interactivity is set to false.

Example

>> Show crosshair based solely on the X axis mouse position.

```
var chart = new EJSC.Chart(
                "chart",
                {show_crosshairs: {x:true,y:false}}
        );
```

5.2.1.13 show_grid

Definition

boolean **show_grid** = **true**

Description

Defines whether to draw the background grid on the chart.

Example

>> Do not draw the background grid

```
var chart = new EJSC.Chart(
                "chart",
                {show_grid: false}
        );
```

5.2.1.14 show_hints

Definition

boolean **show_hints** = true

Description

Defines whether hints should be selected when a point is hovered over or selected. This is automatically disabled if allow_interactivity is set to false.

Example

>> Turn off hints and point selection.

```
var chart = new EJSC.Chart(
                "chart",
                {show_hints: false}
        );
```

5.2.1.15 show_legend

Definition

boolean **show_legend** = true

Description

Defines if the chart legend should be displayed.

Example

>> Turn off legend display

```
var chart = new EJSC.Chart(
                "chart",
                {show_legend: false}
        );
```

5.2.1.16 show_messages

Definition

boolean **show_messages** = true

Description

Defines if progress messages should be displayed above the chart or not.

Example

>> Turn off progress messages.

```
var chart = new EJSC.Chart(
                "chart",
                {show_messages: false}
        );
```

5.2.1.17 show_mouse_position

Definition

boolean **show_mouse_position** = true

Description

Defines if the current mouse position should be displayed above the chart.

Example

>> Turn off mouse position indicators.

```
var chart = new EJSC.Chart(
                "chart",
                {show_mouse_position: false}
        );
```

5.2.1.18 show_titlebar

Definition

boolean **show_titlebar** = true

Description

Defines if the titlebar (containing chart title and mouse position indicators) should be displayed above the chart.

Example

>> Display the chart without a titlebar

```
var chart = new EJSC.Chart(
                "chart",
                {show_titlebar: false}
        );
```

5.2.1.19 show_x_axis

Definition

boolean **show_x_axis** = true

Description

Defines if the X axis (containing x axis caption and tick labels) should be displayed below the chart.

Example

>> Display the chart without the x axis

```
var chart = new EJSC.Chart(
                "chart",
                {show_x_axis: false}
        );
```

5.2.1.20 show_y_axis

Definition

boolean **show_y_axis** = true

Description

Defines if the Y axis (containing y axis caption and tick labels) should be displayed to the left of the chart.

Example

>> Display the chart without the y axis

```
var chart = new EJSC.Chart(
                "chart",
                {show_y_axis: false}
        );
```

5.2.1.21 title

Definition

string **title** = "Emprise JavaScript Chart"

Description

Defines the title of the chart. It is displayed at the top left of the chart.

Example

>> Give the chart a name.

```
var chart = new EJSC.Chart(
                "chart",
                {title: "2007: Total Widget Sales by Month"}
        );
```

5.2.1.22 x_axis_caption

Definition

string **x_axis_caption** = "X Axis"

Description

Defines the text to be displayed below the X axis.

For styling the caption, see EJSC.Chart.x_axis_className

Example

>> Customize the x-axis caption.

```
var chart = new EJSC.Chart(
                "chart",
                {x_axis_caption: "Year"}
        );
```

5.2.1.23 x_axis_className

Definition

string **x_axis_className** = ""

Description

Defines the CSS className to assign to the X-Axis caption.

For styling the tick labels, see EJSC.Chart.x_axis_tick_className

Example

>> Style the X-Axis caption bold.

```
<style> .xAxisCaption { font-style: bold; } </style>

var chart = new EJSC.Chart(
                "chart",
                {x_axis_className: "xAxisCaption"}
        );
```

5.2.1.24 x_axis_extremes_ticks

Definition

boolean **x_axis_extremes_ticks** = false

Description

Defines if the x_min and x_max values should be forced to land on the next tick mark.

Example

>> Force tick marks at the min and max X (left and right sides of the chart)

```
var chart =  new EJSC.Chart(
                "chart",
                {x_axis_extremes_ticks: true}
        );
```

5.2.1.25 x_axis_formatter

Definition

EJSC.Formatter **x_axis_formatter** = EJSC.Formatter

Description

Defines the formatter that will be used to format the tick marks on the X axis before displaying them.

Types:
EJSC.DateFormatter

EJSC.NumberFormatter

Example

>> Display x-axis tick labels as YYYY-MM-DD

```
var chart =  new EJSC.Chart(
               "chart",
               {x_axis_formatter: new EJSC.DateFormatter({format_string: "YYYY-MM-DD"})}
       );
```

5.2.1.26 x_axis_minor_ticks

Definition

object **x_axis_minor_ticks** = {
    show: false,
    color: "rgb(0,0,0)",
    opacity: 20
    thickness: 1,
    count: 7,
    size: 4
}

Description

Defines the properties of the minor tick marks to be drawn on the X axis. The color, opacity and thickness (width of ticks in pixels) properties define the style of the tick marks. The count property defines the number of tick marks to be drawn between each major tick mark. The size property defines the height of the ticks (in pixels).

Example

>> Display red minor tick marks

```
var chart =  new EJSC.Chart(
               "chart",
               {x_axis_minor_ticks: { show: true, color: "#FF0000" }}
       );
```

5.2.1.27 x_axis_size

Definition

integer **x_axis_size** = 20

Description

Defines the height (in pixels) of the X-Axis tick area. To fully enable staggered ticks, set this property to a multiple of 20 (or axis tick height), i.e. two levels = 40, three levels = 60.

For additional control over the format of the labels, see the x_axis_tick_className property.

Example

>> Make X-Axis tick area twice as tall, enabling staggered ticks (default tick label height is 20 pixels)

```
var chart =  new EJSC.Chart(
                "chart",
                { x_axis_size: 40 }
        );
```

5.2.1.28 x_axis_stagger_ticks

Definition

boolean **x_axis_stagger_ticks** = **true**

Description

Determines whether the X-Axis tick labels are staggered.

Example

>> Disable tick staggering

```
var chart =  new EJSC.Chart(
                "chart",
                { x_axis_stagger_ticks: false }
        );
```

5.2.1.29 x_axis_tick_className

Definition

string **x_axis_tick_className** = ""

Description

Defines the CSS className to assign to the X-Axis tick labels. Used in conjunction with x_axis_stagger_ticks and x_axis_size, this property allows for greater control over the size and staggering of tick labels on the X axis.

For styling the caption, see EJSC.Chart.x_axis_className

Example

>> Style the X axis tick labels grey, make them 24 pixels high to enable text wrapping and enable two levels of tick staggering.

```
<style> .xAxisTickLabels { color: #999; height: 24px; } </style>

var chart = new EJSC.Chart(
                "chart",
                {
                        x_axis_tick_className: "xAxisTickLabels",
                        x_axis_size: 48
                }
        );
```

5.2.1.30 x_axis_tick_count

Definition

integer **x_axis_tick_count** = undefined

Description

Defines the number of ticks to be displayed on the X-Axis.

Note: This property is not compatible with text labels (i.e. x axis values are "Gizmos", "Widgets", instead of numbers)

Example

>> Show only 3 ticks on the X-Axis at all times.

```
var chart = new EJSC.Chart(
                "chart",
                {x_axis_tick_count: 3}
        );
```

5.2.1.31 x_cursor_position_caption

Definition

string **x_cursor_position_text** = "X:"

Description

Defines the text to display in front of the current X cursor position in the title area.

Example

>> Change cursor position to display series-related label

```
var chart =  new EJSC.Chart(
                "chart",
                {x_cursor_position_text: "Sales"}
        );
```

5.2.1.32 x_cursor_position_formatter

Definition

EJSC.Formatter **x_cursor_position_formatter** = undefined

Description

Defines the formatter to use for the values displayed in the current X cursor position in the title area. If left undefined, the values will be formatted using the x_axis_formatter.

5.2.1.33 x_max

Definition

float **x_max** = undefined

Description

Defines the current maximum X value of the chart.

Example

>> Force the x-axis range to extend beyond the data it contains

```
var chart =  new EJSC.Chart(
               "chart",
               {x_max: 500.00}
       );
```

5.2.1.34 x_min

Definition

float **x_min** = undefined

Description

Defines the current minimum X value of the chart.

Example

>> Force the x-axis range to extend beyond the data it contains

```
var chart =  new EJSC.Chart(
               "chart",
               {x_min: -100.00}
       );
```

5.2.1.35 x_value_hint_caption

Definition

string **x_value_hint_caption** = "X Value:"

Description

Defines the text to display in front of the X value in the hint (leave blank to hide the X value).

Example

>> Customize the x-value label in the hint window

```
var chart =  new EJSC.Chart(
               "chart",
               {x_value_hint_caption: "Month:"}
       );
```

5.2.1.36 x_zero_plane

Definition

object **x_zero_plane** = {
color: "rgb(0,0,0)",
show: false,
thickness: 1,
coordinate: 0

```
}
```

Description

Defines the properties of the zero plane line to be drawn at X=0 (or whatever the coordinate property is set to). It is used to specify if the line should be shown as well as its color and thickness. The coordinate property allows the base of EJSC.BarSeries with a horizontal orientation to be changed.

Example

>> Display a 2 pixel thick dark green line on the x zero plane

```
var chart =  new EJSC.Chart(
                "chart",
                {x_zero_plane: {show: true, color:'rgb(7,89,5)', thickness:2}}
        );
```

5.2.1.37 y_axis_caption

Definition

string **y_axis_caption** = "Y Axis"

Description

Defines the text to be displayed below the Y axis.

Example

>> Customize the y-axis caption.

```
var chart = new EJSC.Chart(
                "chart",
                {y_axis_caption: "Items Sold"}
        );
```

5.2.1.38 y_axis_className

Definition

string **y_axis_className** = ""

Description

Defines the CSS className to assign to the Y-Axis caption.

For styling the tick labels, see EJSC.Chart.y_axis_tick_className

Example

>> Style the Y-Axis caption bold.

```
<style> .yAxisCaption { font-style: bold; } </style>

var chart = new EJSC.Chart(
                "chart",
                {y_axis_className: "yAxisCaption"}
        );
```

5.2.1.39 y_axis_extremes_ticks

Definition

boolean **y_axis_extremes_ticks** = false

Description

Defines if the y_min and y_max values should be forced to land on the next tick mark.

Example

>> Force tick marks at the min and max Y (bottom and top sides of the chart)

```
var chart =  new EJSC.Chart(
               "chart",
               {y_axis_extremes_ticks: true}
       );
```

5.2.1.40 y_axis_formatter

Definition

EJSC.Formatter **Y_axis_formatter** = EJSC.Formatter

Description

Defines the formatter that will be used to format the tick marks on the Y axis before displaying them.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

Example

>> Display y-axis tick labels as $0.00

```
var chart =  new EJSC.Chart(
               "chart",
               {y_axis_formatter: new EJSC.NumberFormatter({currency_symbol: "$",
forced_decimals: 2, variable_decimals: 2})}
       );
```

5.2.1.41 y_axis_minor_ticks

Definition

object **y_axis_minor_ticks** = {
show: false,
color: "rgb(0,0,0)",
opacity: 20
thickness: 1,
count: 7,

```
        size: 4
}
```

Description

Defines the properties of the minor tick marks to be drawn on the Y axis. The color, opacity and thickness (height of ticks in pixels) properties define the style of the tick marks. The count property defines the number of tick marks to be drawn between each major tick mark. The size property defines the width of the ticks (in pixels).

Example

>> Display red minor tick marks

```
var chart =  new EJSC.Chart(
               "chart",
               {y_axis_minor_ticks: { show: true, color: "#FF0000" }}
       );
```

5.2.1.42 y_axis_size

Definition

integer **y_axis_size** = 50

Description

Defines the width (in pixels) of the Y-Axis tick area.

Example

>> Make Y-Axis tick area half as wide

```
var chart =  new EJSC.Chart(
               "chart",
               { y_axis_size: 25 }
       );
```

5.2.1.43 y_axis_tick_className

Definition

string **y_axis_tick_className** = ""

Description

Defines the CSS className to assign to the Y-Axis tick labels.

For styling the caption, see EJSC.Chart.y_axis_className

Example

>> Style the Y-Axis tick labels grey.

```
<style> .yAxisTickLabels { color: #999; } </style>

var chart = new EJSC.Chart(
```

```
                "chart",
                {y_axis_tick_className: "yAxisTickLabels"}
        );
```

5.2.1.44 y_axis_tick_count

Definition

integer **y_axis_tick_count** = undefined

Description

Defines the number of ticks to be displayed on the Y-Axis.

Example

>> Show only 3 ticks on the Y-Axis at all times.

```
var chart = new EJSC.Chart(
                "chart",
                {y_axis_tick_count: 3}
        );
```

5.2.1.45 y_cursor_position_caption

Definition

string **y_cursor_position_text** = "Y:"

Description

Defines the text to display in front of the current Y cursor position in the title area.

Example

>> Change cursor position to display series-related label

```
var chart =  new EJSC.Chart(
                "chart",
                {y_cursor_position_text: "Price"}
        );
```

5.2.1.46 y_cursor_position_formatter

Definition

EJSC.Formatter **y_cursor_position_formatter** = undefined

Description

Defines the formatter to use for the values displayed in the current Y cursor position in the title area. If left undefined, the values will be formatted using the y_axis_formatter.

5.2.1.47 y_max

Definition

float **y_max** = undefined

Description

Defines the current maximum Y value of the chart.

Example

>> Force the y-axis range to extend beyond the data it contains

```
var chart =  new EJSC.Chart(
               "chart",
               {y_max: 10.00}
       );
```

5.2.1.48 y_min

Definition

float **y_min** = undefined

Description

Defines the current minimum y value of the chart.

Example

>> Force the y-axis range to extend beyond the data it contains

```
var chart =  new EJSC.Chart(
               "chart",
               {y_min: -10.00}
       );
```

5.2.1.49 y_value_hint_caption

Definition

string **y_value_hint_caption** = "Y Value:"

Description

Defines the text to display in front of the Y value in the hint (leave blank to hide the Y value).

Example

>> Customize the y-value label in the hint window

```
var chart =  new EJSC.Chart(
               "chart",
               {y_value_hint_caption: "Cost:"}
       );
```

5.2.1.50 y_zero_plane

Definition

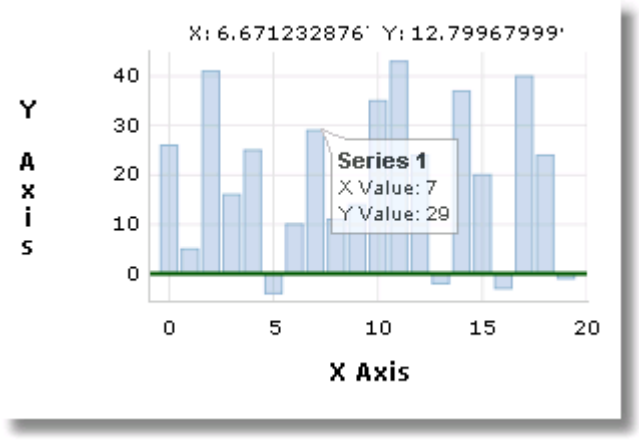
object **y_zero_plane** = {
color: "rgb(0,0,0)",
show: false,
thickness: 1,
coordinate: 0
}

Description

Defines the properties of the zero plane line to be drawn at Y=0 (or whatever the coordinate property is set to). It is used to specify if the line should be shown as well as its color and thickness. The coordinate property allows the base of EJSC.BarSeries and EJSC.AreaSeries to be changed.

Example

>> Display a 2 pixel thick dark green line on the y zero plane




```
var chart =  new EJSC.Chart(
               "chart",
               {y_zero_plane: {show: true, color:'rgb(7,89,5)', thickness:2}}
       );
```

5.2.2 Methods

5.2.2.1 acquireSeries

Definition

void **acquireSeries**(EJSC.Series series, boolean redraw)

Description

Moves a series from one chart to another. This method will remove the series from the chart which currently owns it and add it to the chart making the acquireSeries call.

Sends:
series - The series to be moved

redraw - Boolean flag telling the old and new charts if they should redraw immediately. Specifying **false** here will require that both charts redraw() methods are called in order to display the results of the acquisition.

5.2.2.2 addSeries

Definition

EJSC.Series **addSeries**(EJSC.Series series, boolean redraw)

Description

Adds a new series to the chart and returns the newly added series (this is useful when creating series inline as shown in the example below). The series must descend from EJSC.Series.

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the series to draw. The default, if redraw is omitted is **true**.

Note: If the series has not defined a title (i.e. Series.title = "") at the time addSeries is called, a default title of "Series <index>" will be assigned (where <index> is the current series index in internal series storage).

Types:
EJSC.AnalogGaugeSeries
EJSC.AreaSeries
EJSC.BarSeries
EJSC.FunctionSeries
EJSC.LineSeries
EJSC.PieSeries
EJSC.ScatterSeries
EJSC.TrendSeries

Example

>> Add a line series to the chart.

```
var myChart = new EJSC.Chart("chart");
var mySeries = myChart.addSeries(new EJSC.LineSeries(...));
```

5.2.2.3 addXAxisBin

Definition

void **addXAxisBin**(String label, boolean redraw)

Description

Adds a new static label to the X axis and sets the appropriate flags if necessary to force the chart to draw using static labels (as opposed to a dynamic range based on series data).

This method is useful if there is a need to include bins which may not be part of any series added (i.e. chart labels should be "Apples", "Oranges", "Pears" but the series data only contains data pertaining to Apples and Pears).

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions

such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

Note:

Manually added bins may only be removed if there are no series currently utilizing them.

Example

>> Add bins in a specific order to ensure they appear the same each time the chart is loaded, regardless of the order in which they appear in the data.

```
var myChart = new EJSC.Chart("chart");
myChart.addXAxisBin("Salesman 1");
myChart.addXAxisBin("Salesman 2");
myChart.addXAxisBin("Salesman 3");

var mySeries = myChart.addSeries(
   new EJSC.BarSeries(...)
);
```

5.2.2.4 addYAxisBin

Definition

void **addYAxisBin**(String label, boolean redraw)

Description

Adds a new static label to the Y axis and sets the appropriate flags if necessary to force the chart to draw using static labels (as opposed to a dynamic range based on series data).

This method is useful if there is a need to include bins which may not be part of any series added (i.e. chart labels should be "Apples", "Oranges", "Pears" but the series data only contains data pertaining to Apples and Pears).

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

Note:

Manually added bins may only be removed if there are no series currently utilizing them.

Example

>> Add bins in a specific order to ensure they appear the same each time the chart is loaded, regardless of the order in which they appear in the data.

```
var myChart = new EJSC.Chart("chart");
myChart.addYAxisBin("Salesman 1");
myChart.addYAxisBin("Salesman 2");
myChart.addYAxisBin("Salesman 3");

var mySeries = myChart.addSeries(
   new EJSC.BarSeries(
```

```
        new EJSC.XMLDataHandler(),
        {
            orientation: "horizontal"
        }
    )
);
```

5.2.2.5 clearSelectedPoint

Definition

void **clearSelectedPoint**( )

Description

Clears the current point selection and hides the hint if necessary.

5.2.2.6 convertPixelToPoint

Definition

object **convertPixelToPoint**(integer left, integer top)

Result:

{ x: number, y: number }

Description

Converts the top/left pixel coordinates into chart units. The top/left coordinates sent in should be based on document coordinates, not based off the top left of the chart itself.

X and/or Y may be returned as undefined if the top/left coordinates provided are not within the bounds of the chart.

5.2.2.7 convertPointToPixel

Definition

object **convertPointToPixel**(number x, number y)

Result:

{ left: number, top: number }

Description

Converts the x/y chart coordinates into top/left pixel coordinates based off the document.

Left and/or Top may be returned as undefined if the x/y coordinates provided are not currently within the bounds of the chart.

5.2.2.8 exportSVG

Definition

string **exportSVG**(object Options)

Options Described (parameter {default value})

- **includeHeader {true}**

  The includeHeader property defines whether the result should include the xml declaration (<?xml version="1.0"?>) and the SVG doctype tags.

- **height {chart height in pixels}**

  The height property defines the height attribute of the viewbox attribute in the output svg tag (see viewbox). By default this is the pixel height of the chart but may be set to an arbitrary number or percentage such as "100%"

- **width {chart width in pixels}**

  The width property defines the width attribute of the viewbox attribute in the output svg tag (see viewbox). By default this is the pixel width of the chart but may be set to an arbitrary number or percentage such as "100%"

- **namespace {none}**

  The namespace property defines the prefix namespace to use for the svg tags. This is useful when used with includeHeader=false if including the resulting svg inside another document type. By default, this left undefined and the resulting svg will look something like <svg ... >...</svg>. If defined, for example as namespace = mysvg, the resulting svg will look like <mysvg:svg ...>...</mysvg:svg>.

Description

This method will export the chart as it is currently rendered as SVG. This string can be returned to the server for processing into many different forms such as png or pdf. The server-side processing of this data is beyond the scope of this help document.

Example:

```
<script type="text/javascript">

        var chart = new EJSC.Chart("myChart", {});
        var series = new EJSC.LineSeries(
                new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,1]]),
                {}
        );
        chart.addSeries(series);

        function buttonClick() {
                alert(chart.exportSVG({
                        includeHeader: false,
                        height: "50%",
                        width: "50%",
                        namespace: "mySVG"
                }));
```

```
        }
</script>
<button onclick="buttonClick();">Export SVG</button>
```

5.2.2.9 exportSVGLegend

Definition

string **exportSVGLegend**(object Options)

Options Described (parameter {default value})

- **orientation {"horizontal"}**

  The orientation property defines how the legend dimensions will be calculated. Valid options are "horizontal" and "vertical". When using horizontal, the legend's width will match the chart's width and the legend items will be displayed left to right, top to bottom. The height will expand as needed to accomidate all legend items. Specifying vertical for orientation will cause the legend to match the charts height and display the legend items top to bottom, left to right, expanding the width of the legend as necessary.

- **includeHeader {true}**

  The includeHeader property defines whether the result should include the xml declaration (<?xml version="1.0"?>) and the SVG doctype tags.

- **height {chart height in pixels}**

  The height property defines the height attribute of the viewbox attribute in the output svg tag (see viewbox). By default this is the pixel height of the chart but may be set to an arbitrary number or percentage such as "100%"

- **width {chart width in pixels}**

  The width property defines the width attribute of the viewbox attribute in the output svg tag (see viewbox). By default this is the pixel width of the chart but may be set to an arbitrary number or percentage such as "100%"

- **namespace {none}**

  The namespace property defines the prefix namespace to use for the svg tags. This is useful when used with includeHeader=false if including the resulting svg inside another document type. By default, this left undefined and the resulting svg will look something like <svg ... >...</svg>. If defined, for example as namespace = mysvg, the resulting svg will look like <mysvg:svg ...>...</mysvg:svg>.

- **border {}**

  - **show {true}**

    The show property determines whether or not to draw a border around the entire legend area.

- **color {"#000"}**

  The color property specifies the color of the border, only applicable if show is true.

- **size {1}**

  The size specifies the size (thickness) of the border drawn, only applicable if show is true.

- **show_title {true}**

  The show_title property determines whether to draw the title area of the legend. If false or if the chart's title property is blank, the legend title area will not be drawn.

Description

This method will export the chart legend as it relates to the currently rendered chart. The legend is exported in a more print friendly manner and may be customized using the options described above. The exported SVG string can be returned to the server for processing into many different forms such as png or pdf. The server-side processing of this data is beyond the scope of this help document.

Example:

```
<script type="text/javascript">

        var chart = new EJSC.Chart("myChart", {});
        var series = new EJSC.LineSeries(
                new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,1]]),
                {}
        );
        chart.addSeries(series);

        function buttonClick() {
                alert(chart.exportSVG({
                        height: "50%",
                        width: "50%",
                        border: {
                                show: true,
                                color: "#F00",
                                size: 4
                        },
                        show_title: false
                }));
        }

</script>
<button onclick="buttonClick();">Export SVG</button>
```

5.2.2.10 findClosestPointInSeries

Definition

EJSC.Point **findClosestPointInSeries**(float x, float y, EJSC.Series series)

Description

Returns the closest point to the given coordinates which exists in the given series.

5.2.2.11 getMinMaxYInXRange

Definition

object **getMinMaxYInXRange**(float x_min, float x_max)

RETURNS:
{ y_min: float, y_max: float }

Description

Returns the min and max Y values for the provided X range.

5.2.2.12 getXExtremes

Definition

object **getXExtremes**( )

RETURNS:
{ x_min_extreme: float, x_max_extreme: float }

Description

Returns the current X axis extreme values.

To set the extreme values, see EJSC.Chart.setXExtremes

5.2.2.13 getYExtremes

Definition

object **getYExtremes**( )

RETURNS:
{ y_min_extreme: float, y_max_extreme: float }

Description

Returns the current Y axis extreme values.

To set the extreme values, see EJSC.Chart.setYExtremes

5.2.2.14 getZoom

Definition

object **getZoom**( )

RETURNS:
{ x_min: float, x_max: float, y_min: float, y_max: float }

Description

Returns the current zoom coordinates in chart units.

To set the zoom coordinates, see EJSC.Chart.setZoom

5.2.2.15 getZoomBoxCoordinates

Definition

object **getZoomBoxCoordinates**( )

RETURNS:
{ x_min: float, x_max: float, y_min: float, y_max: float }

Description

Returns the current coordinates of the zoom box (in chart units).

5.2.2.16 hideGrid

Definition

void **hideGrid**(boolean redraw)

Description

Hides the background grid and if redraw is omitted or true, redraws the chart

No effect if the grid is already hidden.

5.2.2.17 hideTitlebar

Definition

void **hideTitlebar**( )

Description

Hides the titlebar, resizes and chart area and redraws all visible series.

No effect if the titlebar is already hidden.

5.2.2.18 hideXAxis

Definition

void **hideXAxis**( )

Description

Hides the X axis, resizes and chart area and redraws all visible series.

No effect if the x axis is already hidden.

5.2.2.19 hideYAxis

Definition

void **hideYAxis**( )

Description

Hides the Y axis, resizes and chart area and redraws all visible series.

No effect if the Y axis is already hidden.

5.2.2.20 redraw

Definition

void **redraw**(boolean reselectPoint)

Description

Resizes chart elements (if necessary) and redraws the entire chart and all series contained within. the **reselectPoint** parameter determines whether to retain point selection (if any) between redraws.

5.2.2.21 remove

Definition

void **remove**( )

Description

Removes the chart from the page, cleans up all attached events, references, timeouts and intervals and clears the contents of the parent element (specified during creation).

5.2.2.22 removeSeries

Definition

void **removeSeries**(EJSC.Series series, boolean redraw)

Description

Removes the specified series from the chart and triggers an immediate rescaling and redraw. Send in redraw = false if performing multiple removes and want to delay redrawing until complete.

5.2.2.23 removeXAxisBin

Definition

void **removeXAxisBin**(String label, boolean redraw)

Description

Removes a bin (static text label) from the X axis.

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

Note:

Bins may only be removed using this method if they were added using addXAxisBin and no series are currently utilizing them.

5.2.2.24 removeYAxisBin

Definition

void **removeYAxisBin**(String label, boolean redraw)

Description

Removes a bin (static text label) from the Y axis.

If redraw is false, drawing will not occur immediately but will not be entirely prevented. Certain actions such as adding an additional series with redraw = true, or chart dimensions changing may still trigger the chart to draw. The default, if redraw is omitted is **true**.

Note:

Bins may only be removed using this method if they were added using addYAxisBin and no series are currently utilizing them.

5.2.2.25 selectClosestPoint

Definition

void **selectClosestPoint**(float x, float y)

Description

Selects the point closest to the coordinates specified.

NOTE: coordinates are in chart coordinates, not pixels

5.2.2.26 selectPoint

Definition

void **selectPoint**(EJSC.Point point, boolean sticky)

Description

Selects the point provided. The sticky property specifies whether the hint window will close when the mouse moves over another point in the chart.

The point object can be retrieved using the Chart.findClosestPointInSeries() method.

5.2.2.27 setCrosshairs

NOT FULLY IMPLEMENTED

5.2.2.28 setLegendTitle

Definition

void **setLegendTitle**(string title)

Description

Updates the caption in the legend window title bar.

5.2.2.29 setTitle

Definition

void **setTitle**(string title)

Description

Updates the chart title shown in the title area of the chart. This method must be used to change the title once the chart has been rendered.

To set a default chart title on creation, see EJSC.Chart.title.

5.2.2.30 setXAxisCaption

Definition

void **setXAxisCaption**(string caption)

Description

Updates the x-axis caption. This method must be used to change the caption once the chart has been rendered. To set a default caption on creation, see EJSC.Chart.x_axis_caption.

5.2.2.31 setXExtremes

Definition

void **setXExtremes**(float x_min, float x_max)

Description

Updates the manual extremes for the x-axis. Use this method if the series data does not span the range to be displayed on the chart or to limit 100% zoom to a range smaller than the series data.

Example

>> Series data only spans 18 hours but the chart needs to display an entire day

```
var myChart = new EJSC.Chart( "chart" );
myChart.setXExtremes( 0, 86400000 );
```

5.2.2.32 setYAxisCaption

Definition

void **setYAxisCaption**(string caption)

Description

Updates the y-axis caption. This method must be used to change the caption once the chart has been rendered. To set a default caption on creation, see EJSC.Chart.y_axis_caption.

5.2.2.33 setYExtremes

Definition

void **setYExtremes**(float y_min, float y_max)

Description

Updates the manual extremes for the y-axis. Use this method if the series data does not span the range to be displayed on the chart or to limit 100% zoom to a range smaller than the series data.

Example

>> Series data only spans 10 to 80 percent but the range displayed should be 0 to 100

```
var myChart = new EJSC.Chart( "chart" );
myChart.setYExtremes( 0, 100 );
```

5.2.2.34 setZoom

Definition:

void **setZoom**(float x_min, float x_max, float y_min, float y_max)

Description

Sets the current zoom of the chart to the incoming parameters.

5.2.2.35 setZoomBoxCoordinates

Definition

voide **setZoomBoxCoordinates**(float x_min, float x_max, float y_min, float y_max)

Description

Displays the zoom box at the given coordinates.

5.2.2.36 showGrid

Definition

void **showGrid**(boolean redraw)

Description

Shows the background grid and if redraw is omitted or true, redraws the chart.

No effect if the grid is already visible.

5.2.2.37 showTitlebar

Definition

void **showTitlebar**( )

Description

Shows the titlebar, resizes and chart area and redraws all visible series.

No effect if the titlebar is already visible.

5.2.2.38 showXAxis

Definition

void **showXAxis**( )

Description

Shows the X axis, resizes and chart area and redraws all visible series.

No effect if the x axis is already visible.

5.2.2.39 showYAxis

Definition

void **showYAxis**( )

Description

Shows the Y axis, resizes and chart area and redraws all visible series.

No effect if the Y axis is already visible.

5.2.3 Events

5.2.3.1 onAfterDraw

Definition

void **onAfterDraw**(EJSC.Chart chart)

Description

Called after a drawing option has completed. Sends the chart object which triggered the event.

5.2.3.2 onAfterMove

Definition

void **onAfterMove**(EJSC.Chart chart)

Description

Called after the chart has been dragged/moved while zoomed in. Sends the chart object which triggered the event.

5.2.3.3 onAfterSelectPoint

Definition

void **onAfterSelectPoint**(EJSC.Point point, EJSC.Series series, EJSC.Chart chart, DOMObject hint_element, string HoverOrSelect)

Description

Called after a point has been selected due to mouse over (hover) or physical selection (click, keyboard).

Sends:
point - The point object selected.
series - The series object in which the point exists.
chart - The chart object which triggered the event (owns the series/point).
hint_element - The DOM object which contains the hint text/markup
HoverOrSelect - String "hover" or "select" to indicate what triggered the event.

5.2.3.4 onAfterShowCrosshairs

Definition

void **onAfterShowCrosshairs**(float x, float y, EJSC.Chart chart)

Description

Called after the crosshairs have been show. The x,y coordinate passed to the event handler represents the current mouse position in chart units.

5.2.3.5 onAfterUnselectPoint

Definition

void **onAfterUnselectPoint**( )

Description

Called after a point has lost selection.

5.2.3.6 onAfterZoom

Definition

void **onAfterZoom**(EJSC.Chart chart)

Description

Called after the chart has been zoomed. Sends the chart object which triggered the event.

5.2.3.7 onBeforeBeginZoom

Definition

boolean **onBeforeBeginZoom**(EJSC.Chart chart, float x, float y)

Description

Called after the beginning mouse position (in chart units) is calculated.

Returning **false** from this event handler will cancel the zoom operation.

5.2.3.8 onBeforeDblClick

Definition

boolean **onBeforeDblClick**(EJSC.Chart chart)

Description

Called before the chart does any processing related to a double click. Return **false** may be used to cancel all additional processing.

5.2.3.9 onBeforeDraw

Definition

boolean **onBeforeDraw**( )

Description

Called before the chart is drawn. If return is false, draw is canceled.

5.2.3.10 onBeforeEndZoom

Definition

boolean **onBeforeEndZoom**(EJSC.Chart chart, float x, float y)

Description

Called after the ending mouse position (in chart units) is calculated and before the actual zoom takes place.

Returning **false** from this event handler will cancel the zoom operation.

5.2.3.11 onBeforeSelectPoint

Definition

boolean **onBeforeSelectPoint**(EJSC.Point point, EJSC.Series series, EJSC.Chart chart, DOMObject hint_element, string HoverOrSelect)

Description

Called after a point has been selected due to mouse over (hover) or physical selection (click, keyboard).

Sends:
point - The point object about to be selected.
series - The series object in which the point exists.
chart - The chart object which triggered the event (owns the series/point).
hint_element - The DOM object which contains the hint text/markup
HoverOrSelect - String "hover" or "select" to indicate what triggered the event.

5.2.3.12 onBeforeUnselectPoint

Definition

void **onBeforeUnselectPoint**(EJSC.Point point)

Description

Called before a point has been unselected. If return is false, current point selection is not lost.

Sends:
point - The point object about to lose selected.

5.2.3.13 onDblClickPoint

Definition

boolean **onDblClickPoint**(EJSC.Point point, EJSC.Series series, EJSC.Chart chart)

Description

Called when a point is double clicked or the ENTER key is pressed and a point is selected. If return is false, cancels zoom-out (if axis are shown).

5.2.3.14 onShowCrosshairs

Definition

void **onShowCrosshairs**(float x, float y, EJSC.Chart chart)

Description

Called when one or both crosshairs are displayed on the chart.

Sends:
x - The x coordinate of the cursor in chart coordinates

y - The y coordinate of the cursor in chart coordinates

5.2.3.15 onShowHint

Definition

string **onShowHint**(EJSC.Point point, EJSC.Series series, EJSC.Chart chart, DOMObject hint_element, string HoverOrSelect)

Description

Called after a point has been selected due to mouse over (hover) or physical selection (click, keyboard).

A custom string may be returned in order to specify the contents of the hint window. See Text Replacement Options for more details.

Sends:
point - The point object selected.
series - The series object in which the point exists.
chart - The chart object which triggered the event (owns the series/point).
hint_element - The DOM object which contains the hint text/markup
HoverOrSelect - String "hover" or "select" to indicate what triggered the event.

5.2.3.16 onShowMessage

Definition

void **onShowMessage**(string message, string type)

Description

Called when the status message is displayed (in the top right corner of the chart).

Sends:
message - The text to be displayed
type - A string identifying the type of message window. This may be "error" or "info"

5.2.3.17 onXAxisNeedsTicks

Definition

array **onXAxisNeedsTicks**(float x_min, float x_max, EJSC.Chart chart)

Description

This event is triggered whenever the axis ticks need to be redrawn / recalculated. It expects an array of [float x, string label] to be returned which defines exactly where to put the tick marks and labels. In addition, null may be returned in order to skip custom ticks for the current draw and use the chart's build in tick controls.

Sends:
x_min - The current minimum x value visible on the chart.
x_max - The current maximum x value visible on the chart.
chart - The chart that triggered the event.

Notes

- To use the label formatter already assigned to the chart, set label to null (i.e. [x_min, null])
- To use on a chart with bins (text instead of numbers), simply send in the bin (i.e. ["First Bin", null])

Example

A typical event handler may look like the following:

```
function doXAxisNeedsTicks(x_min, x_max, chart) {

        // Display 3 tick marks, one at x_min, one at x_max and one directly in between
        var result = new Array();

        result.push( [x_min, null] );
        result.push( [x_min + ((x_max - x_min) / 2), null] );
        result.push( [x_max, null] );

        // Given a chart with a min and max of 0 and 100, the resulting array looks like:
        //  [
        //    [0, null],
        //    [50, null],
        //    [100, null]
        //  ]
        return result;
}
```

5.2.3.18 onYAxisNeedsTicks

Definition

array **onYAxisNeedsTicks**(float y_min, float y_max, EJSC.Chart chart)

Description

This event is triggered whenever the axis ticks need to be redrawn / recalculated. It expects an array of [float y, string label] to be returned which defines exactly where to put the tick marks and labels. In addition, null may be returned in order to skip custom ticks for the current draw and use the chart's build in tick controls.

Sends:

y_min - The current minimum y value visible on the chart.
y_max - The current maximum y value visible on the chart.
chart - The chart that triggered the event.

Notes

- To use the label formatter already assigned to the chart, set label to null (i.e. [y_min, null])
- To use on a chart with bins (text instead of numbers), simply send in the bin (i.e. ["First Bin", null])

Example

A typical event handler may look like the following:

```
function doYAxisNeedsTicks(y_min, y_max, chart) {

        // Display 3 tick marks, one at y_min, one at y_max and one directly in between
        var result = new Array();

        result.push( [y_min, null] );
        result.push( [y_min + ((y_max - y_min) / 2), null] );
        result.push( [y_max, null] );

        // Given a chart with a min and max of 0 and 100, the resulting array looks like:
        //  [
        //    [0, null],
        //    [50, null],
        //    [100, null]
        //  ]
        return result;
}
```

5.3 Series Types

5.3.1 EJSC.AreaSeries

AreaSeries is rendered by drawing a line from point to point and then filling the area defined.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.AreaSeries( EJSC.DataHandler dataHandler [, object options] )
```

Example

```
var mySeries = new EJSC.AreaSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
        { title: "New Area Series" }
);
```

Defining Properties and Events

AreaSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
      var mySeries = new EJSC.AreaSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
      mySeries.lineWidth = 1;
      mySeries.title = "New Area Series";
```

Setting properties in batch

```
      var mySeries = new EJSC.AreaSeries(
             new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
             { lineWidth: 1, title: "New Area Series" }
```

```
);
```

5.3.1.1 Properties

5.3.1.1.1 autosort (inherited)

Definition

boolean **autosort** = true

Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

Note: May not be applicable to all series (i.e. EJSC.PieSeries, EJSC.AnalogGaugeSeries)

5.3.1.1.2 closeLine

Definition

boolean **closeLine** = true

Description

Defines whether the line drawn around the area returns to the zero plane to create a complete shape (true) or if it drawn only through the points in the data set (false).

5.3.1.1.3 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.3.1.1.4 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.3.1.1.5 delayLoad (inherited)

EXPERIMENTAL

Definition

boolean **delayLoad** = true

Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

5.3.1.1.6 drawPoints (inherited)

Definition

boolean **drawPoints** = false

Description

Determines whether the individual points in the series are visually indicated by round dots on the chart. The size and color of the fill and border are determined by the pointColor, pointSize, pointBorderColor and pointBorderSize properties.

Note: Setting this property to true for series which have a large number of points may impact performance as additional draws are required to render the points.

5.3.1.1.7 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.1.1.8 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.1.1.9 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.1.1.10 opacity (inherited)

Definition

integer **opacity** = 50

Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setOpacity()

5.3.1.1.11 pointBorderColor (inherited)

Definition

string **pointBorderColor** = "rgb(255,255,255)"

Description

Defines the color of the border drawn around the points. This property is only applicable when the series drawPoints property is set to true and pointBorderSize is greater than 0.

5.3.1.1.12 pointBorderSize (inherited)

Definition

integer **pointBorderSize** = 0

Description

Defines the size in pixels of the border drawn around the points. This property is only applicable when the series drawPoints property is set to true.

To draw points without any border, leave pointBorderSize set to 0.

5.3.1.1.13 pointColor (inherited)

Definition

string **pointColor** = undefined

Description

Defines the color of the point (circle) drawn at each data point in the series. If left undefined the point will inherit the series color.

This property is only applicable when the series drawPoints property is set to true.

5.3.1.1.14 pointSize (inherited)

Definition

integer **pointSize** = undefined

Description

Defines the size of the point (circle) drawn at each data point in the series. If left undefined the point diameter will be twice the line width of the series.

This property is only applicable when the series drawPoints property is set to true.

5.3.1.1.15 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.1.1.16 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.1.1.17 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter

EJSC.NumberFormatter

5.3.1.1.18 y_axis_formatter (inherited)

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.1.2 Methods

5.3.1.2.1 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.3.1.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.1.2.3 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.1.2.4 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.1.2.5 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.1.2.6 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.1.2.7 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.3.1.2.8 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.3.1.2.9 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.1.2.10 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

5.3.1.2.11 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.1.2.12 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.1.2.13 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.1.3 Events

5.3.1.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.3.1.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.1.3.3 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.2 EJSC.AnalogGaugeSeries

In order to use the AnalogGaugeSeries, the file EJSChart_Gauges.js must be included in the HTML page after the inclusion of the standard EJSChart.js file.

```
<head>
        <script type="text/javascript" src="/EJSChart/EJSChart.js"></script>
        <script type="text/javascript" src="/EJSChart/EJSChart_Gauges.js"></script>
</head>
```

The AnalogGaugeSeries is rendered as a circular (or semi-circular) gauge with a needle directed towards the current value stored in its EJSC.GaugePoint.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.AnalogGaugeSeries( EJSC.DataHandler dataHandler [, object options] )
```

Example

```
var myAnalogGaugeSeries = new EJSC.AnalogGaugeSeries(
    new EJSC.ArrayDataHandler([[50,"Gauge Value"]]),
    { title: "New Analog Gauge Series" }
);
```

Defining Properties and Events

AnalogGaugeSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var mySeries = new EJSC.AnalogGaugeSeries(new EJSC.ArrayDataHandler([[50,"Gauge
Value"]]));
mySeries.title = "New Analog Gauge Series";
mySeries.min = 0;
mySeries.max = 100;
```

Setting properties in batch

```
var mySeries = new EJSC.AnalogGaugeSeries(
    new EJSC.ArrayDataHandler([[50,"Gauge Value"]]),
    { title: "New Analog Gauge Series", min: 0, max: 100 }
);
```

5.3.2.1 Properties

5.3.2.1.1 anchor

Definition

```
object anchor = {
    color:      'rgb(0,0,0)' ,
    opacity:    100 ,
    size:       10
}
```

Description

Defines the parameters for the anchor on the gauge.

Properties

string **color** – Defines the color of the anchor.
integer **opacity** – Defines the opacity of the anchor.
integer **size** – Defines the diameter (in pixels) of the anchor.

5.3.2.1.2 axis

Definition

```
object axis = {
        color:              'rgb(255,255,255)' ,
        innerBorderColor:   'rgb(0,0,0)' ,
        innerBorderOpacity: 100 ,
        innerBorderWidth:   1 ,
        innerBorderVisible: true ,
        opacity:            0 ,
        outerBorderColor:   'rgb(0,0,0)' ,
        outerBorderOpacity: 100 ,
        outerBorderWidth:   1 ,
        outerBorderVisible: true ,
        thickness:          15
}
```

Description

Defines the parameters for the axis on the gauge.

Properties

string	**color**	– Defines the background color of the axis.
string	**innerBorderColor**	– Defines the color of the inner border.
integer	**innerBorderOpacity**	– Defines the opacity of the inner border.
integer	**innerBorderWidth**	– Defines the width of the inner border.
integer	**innerBorderVisible**	– Defines whether the inner border is visible or not.
integer	**opacity**	– Defines the opacity of the background color of the axis.
string	**outerBorderColor**	– Defines the color of the outer border.
integer	**outerBorderOpacity**	– Defines the opacity of the outer border.
integer	**outerBorderWidth**	– Defines the width of the outer border.
boolean	**outerBorderVisible**	– Defines whether the outer border is visible or not.
integer	**thickness**	– Defines the thickness of the axis.

5.3.2.1.3 delayLoad (inherited)

EXPERIMENTAL

Definition

boolean **delayLoad** = true

Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

5.3.2.1.4 fillColor

Definition

string **fillColor** = undefined

Description

Defines the background fill color of the gauge. Leave undefined for no fill color. This should be defined as rgb(<RED>,<GREEN>,<BLUE>), i.e. red = "rgb(255,0,0)"

5.3.2.1.5 fillOpacity

Definition

integer **fillOpacity** = 100

Description

Defines the opacity for the background fill color of the gauge.

5.3.2.1.6 height

Definition

string **height** = "100%"

Description

Defines the total height of the gauge. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. height = 150;).

5.3.2.1.7 label

Definition

```
object label = {
        className:      '' ,
        textAlign:      'center' ,
        position:       'centerBottom' ,
        lines:          1
}
```

Description

Defines the parameters for the label on the gauge.

Properties

string **className** – Defines a class (to be defined in an attached stylesheet) to be applied to the label.
string **textAlign** – Defines the alignment of the text in the label ('left', 'center', or 'right').
string **position** – Defines the position on the gauge the label will be placed (see below).
integer **lines** – Defines the number of lines to display in the label. (Text will overflow unless otherwise specified in the attached style class).

Values for position:

- bottom
- centerBottom
- centerLeft
- centerRight

- centerTop
- left
- right
- top

5.3.2.1.8 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.2.1.9 lock

Definition

```
object lock = {
        color:      'rgb(0,0,0)' ,
        offset:     5 ,
        opacity:    100 ,
        size:       6 ,
        visible:    false
}
```

Description

Defines the parameters for the "locks", or "pegs" on the gauge.

Properties

string **color** – Defines the color of the locks.
integer **offset** – Defines the distance (in pixels) the needle is allow to extend past the min/max value. **Set to undefined to turn locks off.**
integer **opacity** – Defines the opacity of the locks.
integer **size** – Defines the diameter (in pixels) of the locks.
boolean **visible** – Defines whether or not to visually show the locks.

5.3.2.1.10 marker_position

Definition

string **marker_position** = "outer"

Description

Defines where the tick markers will be positioned with respect to the axis.

Quadrants:

- outer
- inner

5.3.2.1.11 max

Definition

float **max** = 100

Description

Defines the maximum value to be displayed on the gauge.

5.3.2.1.12 min

Definition

float **min** = 0

Description

Defines the minimum value to be displayed on the gauge.

5.3.2.1.13 minorTick

Definition

```
object minorTick = {
        color: 'rgb(150,150,150)' ,
        count: 4 ,
        offset: 0 ,
        opacity: 100 ,
        size:  1 ,
        thickness:  15
}
```

Description

Defines the parameters for the minor ticks on the gauge.

Properties

string **color** – Defines the color of the minor tick marks.
integer **count** – Defines the number of minor tick marks to display between each major tick mark.
integer **offset** – Defines the distance (in pixels) the minor tick marks are pushed out from the inner axis border.
integer **opacity** – Defines the opacity of the minor tick marks.
integer **size** – Defines the width (in pixels) of the minor tick marks.
integer **thickness** – Defines the thickness (in pixels) of the minor tick marks.

5.3.2.1.14 needle

Definition

```
object needle = {
        borderColor:    'rgb(0,0,0)' ,
        borderOpacity:  100 ,
        borderWidth:    1 ,
        color:          'rgb(0,0,0)' ,
        opacity:        100 ,
        size:           4
}
```

Description

Defines the parameters for the needle on the gauge.

Properties

string	**borderColor**	– Defines the color of the border.
integer	**borderOpacity**	– Defines the opacity of the border.
integer	**borderWidth**	– Defines the width of the border.
string	**color**	– Defines the background color of the needle.
integer	**opacity**	– Defines the opacity of the needle.
integer	**size**	– Defines the width (in pixels) of the needle at its base.

5.3.2.1.15 position

Definition

string **position** = "center"

Description

Defines the quadrant of the chart that the gauge will appear in.

Quadrants:

- topLeft
- topCenter
- topRight
- leftCenter
- center
- rightCenter
- bottomLeft
- bottomCenter
- bottomRight

5.3.2.1.16 range

Definition

```
object range = {
        borderColor:    'rgb(0,0,0)' ,
        borderWidth:    1 ,
        offset:         0 ,
        opacity:        100 ,
        style:          'doughnut' ,
        thickness:      15
```

}

Description

Defines the parameters for the ranges on the gauge.

Properties

string	**borderColor**	– Defines the color of the border.
integer	**borderWidth**	– Defines the width of the border.
integer	**offset**	– Defines the distance (in pixels) the ranges are pushed in from the axis.
integer	**opacity**	– Defines the opacity of the ranges.
string	**style**	– Defines the style to draw the range in ('doughnut' or 'pie')
integer	**thickness**	– Defines the thickness (in pixels) of the ranges.

5.3.2.1.17 range_degrees

Definition

integer **range_degrees** = 180

Description

Defines the total angle (in degrees) the gauge will take up.

5.3.2.1.18 ranges

Definition

array **ranges** = new Array();

Description

Defines a series of ranges to be marked in the gauge.

Implementation

mySeries.ranges = [[int min, int max, string color] , ...]

Example

```
mySeries.ranges = [
   [0,10,'rgb(255,0,0)'],
   [10,20,'rgb(0,255,0)']
];
```

5.3.2.1.19 start_degree

Definition

integer **start_degree** = 270

Description

Defines the angle (in degrees) at which the gauges' min value is displayed.

5.3.2.1.20 tick

Definition

object **tick** = {
    className: " ,
    color: 'rgb(0,0,0)' ,
    offset: 0 ,
    opacity: 100 ,
    size: 1 ,
    thickness: 15
}

Description

Defines the parameters for the major ticks on the gauge.

Properties

string **className** – Defines a class (to be defined in an attached stylesheet) to be applied to the tick markers.
string **color** – Defines the color of the tick marks.
integer **offset** – Defines the distance (in pixels) the tick marks are pushed out from the inner axis border.
integer **opacity** – Defines the opacity of the tick marks.
integer **size** – Defines the width (in pixels) of the tick marks.
integer **thickness** – Defines the thickness (in pixels) of the tick marks.

5.3.2.1.21 tickCount

Definition

integer **tickCount** = 11

Description

Defines the number of major ticks to be displayed on the gauge's axis

5.3.2.1.22 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.2.1.23 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.2.1.24 width

Definition

string **width** = "100%"

Description

Defines the total width of the gauge. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. width = 150;).

5.3.2.1.25 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.2.2 Methods

5.3.2.2.1 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.3.2.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.2.2.3 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.2.2.4 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.2.2.5 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.2.2.6 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.3.2.2.7 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.2.2.8 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.2.2.9 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.2.3 Events

5.3.2.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.3.2.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.2.3.3 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.3 EJSC.BarSeries

BarSeries renders its points as vertical bars.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.BarSeries( EJSC.DataHandler dataHandler [, object options] )
```

Example

```
var mySeries = new EJSC.BarSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
        { title: "New Bar Series" }
);
```

Defining Properties and Events

BarSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var mySeries = new EJSC.BarSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.lineWidth = 1;
mySeries.title = "New Bar Series";
```

Setting properties in batch

```
var mySeries = new EJSC.BarSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
```

```
                { lineWidth: 1, title: "New Bar Series" }
        );
```

5.3.3.1 Properties

5.3.3.1.1 autosort (inherited)

Definition

boolean **autosort** = true

Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

Note: May not be applicable to all series (i.e. EJSC.PieSeries, EJSC.AnalogGaugeSeries)

5.3.3.1.2 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.3.3.1.3 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.3.3.1.4 defaultColors

Definition

array **defaultColors** = EJSC.DefaultBarColors

Description

Defines the pool of default colors available for bar series bars.

NOTE: To make use of individually colored bars, the EJSC.BarSeries.useColorArray property must be set to true.

5.3.3.1.5 delayLoad (inherited)

EXPERIMENTAL

Definition

boolean **delayLoad** = true

Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

5.3.3.1.6 groupedBars

Definition

boolean **groupedBars** = true

Description

Defines whether the bars from different bar series in a single chart are grouped or overlayed.

NOTE: Only the first bar series added to the chart may use this property to affect the grouped/overlay display. To change the style after the first bar series has been added, use the EJSC.BarSeries.setGroupedBars from any bar series in the chart. Currently you cannot mix both overlayed and grouped bars within the same chart.

Example

>> Make the bar series overlay

```
var chart =  new EJSC.Chart("chart");
var bar1 = chart.addSeries(new EJSC.BarSeries(
   data,
   {
      groupedBars: false,
   }
));
var bar2 = chart.addSeries(new EJSC.BarSeries(data));
```

5.3.3.1.7 intervalOffset

Definition

float **intervalOffset** = 0.8

Description

Defines the spacing between the bars as a percentage of 1. 1 = bars take up the entire width available, with their sides touching.

Example

>> Make the bars take up 1/2 their available width. (i.e. more space between bars than by default)

```
var chart =  new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
   data,
   {
      intervalOffset: 0.5
   }
));
```

5.3.3.1.8 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.3.1.9 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.3.1.10 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.3.1.11 opacity (inherited)

Definition

integer **opacity** = 50

Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setOpacity()

5.3.3.1.12 orientation

Definition

string **orientation** = "vertical"

Description

This property determines the orientation of the bar series. The supported property values are "vertical" and "horizontal".

Note: Currently this is a creation-time only property and should not be changed after the series has been created.

5.3.3.1.13 ranges

Definition

array **ranges** = new Array()

Description

This property is used to store the ranges for a given bar series. The range objects are used to draw bars in varying styles based on their Y value.

The range objects stored are defined as:

```
range = {
        min:        float, // >=
        max:        float, // <
        color:      string, // Defined as rgb(RED,GREEN,BLUE), ex: "rgb(255,0,0)"
        opacity:    integer, // Represents percent, i.e. 50 = 50%
        lineOpacity: integer, // Represents percent, i.e. 50 = 50%
        lineWidth:  integer // Represents the line width in pixels
}
```

Example

>> Define ranges so that bars with a Y value from 0 to 10 and 90 to 100 are colored red, bars 10 - 90 are colored green

```
var chart =  new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
   data,
   {
      ranges: [
         { 0, 10, "rgb(255,0,0)", 100, 100, 1 },
         { 90, 100, "rgb(255,0,0)", 100, 100, 1 },
         { 10, 90, "rgb(0,255,0)", 100, 100, 1 }
      ]
   }
));
```

5.3.3.1.14 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.3.1.15 treeLegend

Definition

boolean **treeLegend** = false

Description

Defines whether to display each bar individually in the legend.

If the useColorArray property is true and treeLegend left at its default (not defined in options), the treeLegend property will automatically be set to true to enable tree-style legend display.

5.3.3.1.16 useColorArray

Definition

boolean **useColorArray** = false

Description

Defines whether to make use of the EJSC.BarSeries.defaultColors property

Example

>> Use the color array to make a chart of multi colored bars

```
var chart =  new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
   data,
   {
      useColorArray: true
   }
));
```

5.3.3.1.17 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.3.1.18 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.3.1.19 y_axis_formatter (inherited)

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.3.2 Methods

5.3.3.2.1 addRange

Definition

void **addRange**(float min, float max, string color, integer opacity, integer lineOpacity, integer lineWidth, boolean redraw)

Description

Adds a new range to the bar series and optionally triggers a redraw.

5.3.3.2.2 clearRanges

Definition

void **clearRanges**(boolean redraw)

Description

Clears all ranges defined for the series and optionally redraws the the chart.

5.3.3.2.3 deleteRange

Definition

void **deleteRange**(float min, float max, boolean redraw)

Description

Deletes the matching range (min and max must exactly match an existing range) and optionally redraws the chart.

5.3.3.2.4 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.3.3.2.5 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.3.2.6 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.3.2.7 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.3.2.8 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.3.2.9 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.3.2.10 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.3.3.2.11 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.3.3.2.12 setDefaultColors

Definition

void **setDefaultColors**(array colors, boolean reload)

Description

Set the array of default colors available for bar series bars.

If reload = **true** the bars (if loaded) will pull their colors from the given color array and the chart will be redrawn.

If onBarNeedsColor is assigned, this method will ignore that event and load the colors from the array.

Example

```
var colors = [
        "rgb('0,0,0')",         // black
        "rgb(255,0,0)",         // red
        "rgb("0,255,0)",        // green
        "rgb("0,0,255)",        // blue
        "rgb("255,255,255)"     // white
];

myBarSeries.setDefaultColors(colors, true);
```

5.3.3.2.13 setGroupedBars

Definition

void **setGroupedBars**(boolean grouped, boolean redraw)

Description

Changes the chart between grouped and overlayed bars and optionally triggers a redraw.

5.3.3.2.14 setIntervalOffset

Definition

void **setIntervalOffset**(float offset, boolean redraw)

Description

Updates the interval offset property (spacing between the bars) and optionally redraws the chart. See EJSC.BarSeries.intervalOffset for additional information.

5.3.3.2.15 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.3.2.16 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

5.3.3.2.17 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.3.2.18 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.3.2.19 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.3.3 Events

5.3.3.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.3.3.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.3.3.3 onBarNeedsColor

Definition

string **onBarNeedsColor**(EJSC.BarPoint point, EJSC.Series series, EJSC.Chart chart)
object **onBarNeedsColor**(EJSC.BarPoint point, EJSC.Series series, EJSC.Chart chart)

Description

Fired when a bar needs a color. This event may return either a string (i.e. "rgb(0,255,0)") or an object with additional styling properties.

The object returned is expected to be formatted as:

```
{
        color:          string,
        opacity:        integer,
        lineOpacity:    integer,
        lineWidth:      integer
}
```

Example

>> Display bars with userdata of "bold" with thicker lines

```
var chart = new EJSC.Chart("chart");
var bar = chart.addSeries(new EJSC.BarSeries(
   data,
   {
      lineOpacity: 80,
      lineWidth: 1,
      onBarNeedsColor: function(point, series, chart) {

         if (point.userdata == "bold") {
            return {
               color: "rgb(0,0,0)",
               opacity: 100,
               lineOpacity: 100,
               lineWidth: 100
            };
         } else {
            return "rgb(128,128,128)";
         }
      }
   }
));
```

5.3.3.3.4 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.4 EJSC.FunctionSeries

The FunctionSeries is rendered as a line based on the results of the function specified.

The constructor expects a function which can be called as function(x), and an optional set of object properties.

Constructor

EJSC.FunctionSeries(function fn [, object options])

Example

```
var mySeries = new EJSC.FunctionSeries(
        Math.cos,
        { title: "New Function Series" }
);
```

Defining Properties and Events

FunctionSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var mySeries = new EJSC.FunctionSeries(Math.cos);
mySeries.lineWidth = 2;
mySeries.title = "New Function Series";
```

Setting properties in batch

```
var mySeries = new EJSC.FunctionSeries(
        Math.cos,
        { lineWidth: 2, title: "New Function Series" }
);
```

5.3.4.1 Properties

5.3.4.1.1 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.3.4.1.2 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.3.4.1.3 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.4.1.4 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.4.1.5 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.4.1.6 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.4.1.7 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.4.1.8 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.4.1.9 y_axis_formatter (inherited)

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left

undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.4.2 Methods

5.3.4.2.1 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.4.2.2 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.4.2.3 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.4.2.4 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.4.2.5 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.4.2.6 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.3.4.2.7 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.4.2.8 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.4.2.9 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.4.2.10 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.4.3 Events

5.3.4.3.1 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.4.3.2 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.5 EJSC.LineSeries

LineSeries is rendered by drawing a line from point to point.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.LineSeries( EJSC.DataHandler dataHandler [, object options] )
```

Example

```
var mySeries = new EJSC.LineSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
```

```
        { title: "New Line Series" }
);
```

Defining Properties and Events

LineSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var mySeries = new EJSC.LineSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
mySeries.lineWidth = 2;
mySeries.title = "New Line Series";
```

Setting properties in batch

```
var mySeries = new EJSC.LineSeries(
        new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
        { lineWidth: 2, title: "New Line Series" }
);
```

5.3.5.1 Properties

5.3.5.1.1 autosort (inherited)

Definition

boolean **autosort** = true

Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

Note: May not be applicable to all series (i.e. EJSC.PieSeries, EJSC.AnalogGaugeSeries)

5.3.5.1.2 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.3.5.1.3 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.3.5.1.4 drawPoints

Definition

boolean **drawPoints** = false

Description

Determines whether the individual points in the series are visually indicated by round dots on the chart. The size and color of the fill and border are determined by the pointColor, pointSize, pointBorderColor and pointBorderSize properties.

Note: Setting this property to true for series which have a large number of points may impact performance as additional draws are required to render the points.

5.3.5.1.5 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.5.1.6 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.5.1.7 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.5.1.8 pointBorderColor

Definition

string **pointBorderColor** = "rgb(255,255,255)"

Description

Defines the color of the border drawn around the points. This property is only applicable when the series drawPoints property is set to true and pointBorderSize is greater than 0.

5.3.5.1.9 pointBorderSize

Definition

integer **pointBorderSize** = 0

Description

Defines the size in pixels of the border drawn around the points. This property is only applicable when the series drawPoints property is set to true.

To draw points without any border, leave pointBorderSize set to 0.

5.3.5.1.10 pointColor

Definition

string **pointColor** = undefined

Description

Defines the color of the point (circle) drawn at each data point in the series. If left undefined the point will inherit the series color.

This property is only applicable when the series drawPoints property is set to true.

5.3.5.1.11 pointSize

Definition

integer **pointSize** = undefined

Description

Defines the size of the point (circle) drawn at each data point in the series. If left undefined the point diameter will be twice the line width of the series.

This property is only applicable when the series drawPoints property is set to true.

5.3.5.1.12 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.5.1.13 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.5.1.14 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.5.1.15 y_axis_formatter (inherited)

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.5.2 Methods

Enter topic text here.

5.3.5.2.1 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.3.5.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.5.2.3 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.5.2.4 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.5.2.5 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data

again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.5.2.6 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.5.2.7 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.3.5.2.8 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.3.5.2.9 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.5.2.10 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.5.2.11 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.5.2.12 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.5.3 Events

Enter topic text here.

5.3.5.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.3.5.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.5.3.3 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.6 EJSC.PieSeries

PieSeries is rendered by drawing slices to form an ellipse. Each slice represents a percentage of the total of the sum of all point values in the dataset.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.PieSeries( EJSC.DataHandler dataHandler [, object options] )
```

Example

```
var mySeries = new EJSC.PieSeries(
        new EJSC.ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label3"]]),
        { title: "New Pie Series" }
);
```

Defining Properties and Events

PieSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var mySeries = new EJSC.PieSeries(new EJSC.ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label 3"]]));
mySeries.title = "New Pie Series";
```

Setting properties in batch

```
var mySeries = new EJSC.PieSeries(
        ArrayDataHandler([[10,"Label 1"],[20,"Label 2"],[120,"Label 3"]]),
        { title: "New Pie Series" }
);
```

5.3.6.1 Properties

5.3.6.1.1 defaultColors

Definition

array **defaultColors** = EJSC.DefaultPieColors

Description

Defines the pool of default colors available for pie pieces.

5.3.6.1.2 delayLoad (inherited)

EXPERIMENTAL

Definition

boolean **delayLoad** = true

Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

5.3.6.1.3 height

Definition

string **height** = "100%"

Description

Defines the total height of the pie series. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. height = 150).

5.3.6.1.4 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.6.1.5 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.6.1.6 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.6.1.7 opacity (inherited)

Definition

integer **opacity** = 50

Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setOpacity()

5.3.6.1.8 position

Definition

string **position** = "center"

Description

Defines the quadrant of the chart that the pie will appear in.

Quadrants:

- topLeft
- topCenter
- topRight
- leftCenter
- center
- rightCenter
- bottomLeft
- bottomCenter
- bottomRight

5.3.6.1.9 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array

for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.6.1.10 total_value

Definition

integer **total_value** = undefined

Description

Defines the total value (sum) of all pie pieces in the series. If left undefined, this value will be calculated automatically by adding all the piece values when the series data is loaded.

This property is only applicable during series creation. To determine or modify the total value once the series has been created use the getTotalValue and setTotalValue methods.

To force the chart to recalculate the total value based on actual series data, use the resetTotalValue method.

5.3.6.1.11 treeLegend

Definition

boolean **treeLegend** = true

Description

Defines whether to display each pie piece individually in the legend.

5.3.6.1.12 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.6.1.13 width

Definition

string **width** = "100%"

Description

Defines the total width of the pie series. This may be specified in percent of the chart as shown above as the default value, or in exact pixels by specifying a number (i.e. width = 150).

5.3.6.1.14 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.6.2 Methods

5.3.6.2.1 findCenter

Definition

object **findCenter**(EJSC.PiePoint point)

The object returned is formatted as:

```
{
  x: integer,
  y: integer
}
```

Description

This method takes a pie point (see getPoints) and returns an object containing the x and y screen coordinates of the piece's center.

5.3.6.2.2 findCenterOfCurve

Definition

object **findCenterOfCurve**(EJSC.PiePoint point)

The object returned is formatted as:

```
{
  x: integer,
  y: integer
}
```

Description

This method takes a pie point (see getPoints) and returns an object containing the x and y screen coordinates of the center of the piece's arc.

5.3.6.2.3 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.3.6.2.4 getPoints

Definition

array **getPoints**()

Description

This method returns an array of all EJSC.PiePoint objects in the series.

5.3.6.2.5 getTotalValue

Definition

integer **getTotalValue**( )

Description

Returns the current total value of the pie series. If the total_value property was set during creation or the setTotalValue has been called, the static value will be returned. If the series is set to calculate the total value based on the series data, the dynamic value will be returned.

5.3.6.2.6 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.6.2.7 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.6.2.8 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.6.2.9 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.6.2.10 resetTotalValue

Definition

void **resetTotalValue**(boolean redraw)

Description

This method causes the series to recalculate its total value based on the actual series data and optionally redraws the series to reflect the change.

To retrieve the total value after this method is called, use getTotalValue.

5.3.6.2.11 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.3.6.2.12 setDefaultColors

Definition

void **setDefaultColors**(array colors, boolean reload)

Description

Set the array of default colors available for pie pieces.

If reload = **true** the pie pieces (if loaded) will pull their colors from the given color array and the chart will be redrawn.

If onPieceNeedsColor is assigned, this method will ignore that event and load the colors from the array.

Example

```
var colors = [
        "rgb('0,0,0')",         // black
        "rgb(255,0,0)",         // red
        "rgb("0,255,0)",        // green
        "rgb("0,0,255)",        // blue
        "rgb("255,255,255)"     // white
];

myPieSeries.setDefaultColors(colors, true);
```

5.3.6.2.13 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.6.2.14 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

5.3.6.2.15 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.6.2.16 setTotalValue

Definition

void **setTotalValue**(integer value, boolean redraw)

Description

Sets the total value of the pie series and optionally redraws the series to reflect the change.

To force the chart to calculate its total value based on the actual series data, call resetTotalValue.

5.3.6.2.17 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.6.2.18 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.6.3 Events

5.3.6.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.3.6.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.6.3.3 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.6.3.4 onPieceNeedsColor

Definition

boolean **onPieceNeedsColor**(EJSC.PiePoint point, EJSC.PieSeries series, EJSC.Chart chart)

Description

If assigned, this event is triggered for each piece in a pie series immediately before it pulls a color from the default colors array. The event provides the EJSC.PiePoint object which represents the piece which needs a color, the EJSC.PieSeries which owns the piece and the EJSC.Chart which owns the series. Return a color string formatted as "rgb(<red value>, <green value>, <blue value>)", i.e. Orange would be "rgb(237,173,0)"

5.3.7 EJSC.ScatterSeries

ScatterSeries is rendered by drawing a styled point for each x,y coordinate in the dataset.

The constructor expects an instantiated EJSC.DataHandler descendant and an optional set of object properties.

Constructor

```
EJSC.ScatterSeries( EJSC.DataHandler dataHandler [, object options] )
```

Example

```
var mySeries = new EJSC.ScatterSeries(
    new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
    { title: "New Scatter Series" }
);
```

Defining Properties and Events

ScatterSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
    var mySeries = new EJSC.ScatterSeries(new
EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]));
    mySeries.pointSize = 3;
```

```
mySeries.pointStyle = "diamond";
mySeries.title = "New Scatter Series";
```

Setting properties in batch

```
var mySeries = new EJSC.ScatterSeries(
       new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,3],[5,2],[6,4]]),
       { pointSize: 3, pointStyle: "diamond", title: "New Scatter Series" }
);
```

5.3.7.1 Properties

5.3.7.1.1 autosort (inherited)

Definition

boolean **autosort** = true

Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

Note: May not be applicable to all series (i.e. EJSC.PieSeries, EJSC.AnalogGaugeSeries)

5.3.7.1.2 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.3.7.1.3 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.3.7.1.4 delayLoad (inherited)

EXPERIMENTAL

Definition

boolean **delayLoad** = true

Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

5.3.7.1.5 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.3.7.1.6 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.7.1.7 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.7.1.8 opacity (inherited)

Definition

integer **opacity** = 50

Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setOpacity()

5.3.7.1.9 pointSize

Definition

integer **pointSize** = 4

Description

Defines the size of the point to be drawn. Point size has no effect on point selection. In order to increase the distance from the actual coordinate that a click will select or hover a point, see EJSC.Chart.proximity_snap

5.3.7.1.10 pointStyle

Definition

string **pointStyle** = "triangle"

Styles:
- triangle
- box
- circle
- diamond

Description

Defines the shape of the point to be drawn. Point shape has no effect on point selection. In order to increase the distance from the actual coordinate that a click will select or hover a point, see EJSC.Chart.proximity_snap

5.3.7.1.11 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.7.1.12 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.7.1.13 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.7.1.14 y_axis_formatter (inherited)

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.7.2 Methods

5.3.7.2.1 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.3.7.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.7.2.3 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.7.2.4 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.7.2.5 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.7.2.6 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.7.2.7 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.3.7.2.8 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.3.7.2.9 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.7.2.10 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

5.3.7.2.11 setPointStyle

Definition

void **setPointStyle**(integer size, string style)

Description

Updates the pointSize and/or pointStyle and redraws the chart. Size or style update may be avoided by sending undefined.

See EJSC.ScatterSeries.pointStyle for a list of available styles.

Example

>> Update the size and style of scatter series points

```
mySeries.setPointStyle( 5, "diamond" );
```

>> Update only the style of scatter series points

mySeries.setPointStyle(undefined, "box");

5.3.7.2.12 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.7.2.13 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.7.2.14 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.7.3 Events

5.3.7.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.3.7.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.7.3.3 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.3.8 EJSC.TrendSeries

The TrendSeries is rendered as a line based on the results of a function applied to the related series' data points.

The constructor expects a reference series, a `trendType` and an optional set of object properties.

`The trend series then “trends” the data in that referenceSeries according to the trendType into a function, and plots the function in the visible area of the chart.`

Constructor

EJSC.TrendSeries(EJSC.Series referenceSeries, string trendType [, object options])

Valid options for trendType:
"linear"
"power"
"exponential"
"logarithmic"

Example

```
var mySeries1 = new EJSC.ScatterSeries(...);
var mySeries2 = new EJSC.TrendSeries(
        mySeries1,
        "linear",
        { title: "New Trend Series" }
);
```

Defining Properties and Events

TrendSeries properties may be set individually after the series has been created or in batch using the options parameter during instantiation.

Setting properties individually

```
var mySeries = new EJSC.TrendSeries(myDataSeries, "linear");
mySeries.lineWidth = 2;
mySeries.title = "New Trend Series";
```

Setting properties in batch

```
var mySeries = new EJSC.TrendSeries(
        myDataSeries,
        "linear",
        { lineWidth: 2, title: "New Trend Series" }
);
```

5.3.8.1 Properties

5.3.8.1.1 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.3.8.1.2 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.3.8.1.3 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series

creation.

5.3.8.1.4 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.3.8.1.5 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.8.1.6 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.3.8.1.7 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.3.8.1.8 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.8.1.9 y_axis_formatter (inherited)

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.3.8.2 Methods

5.3.8.2.1 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.8.2.2 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.3.8.2.3 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.3.8.2.4 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.3.8.2.5 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.8.2.6 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.3.8.2.7 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.8.2.8 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.3.8.2.9 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.3.8.2.10 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.3.8.3 Events

5.3.8.3.1 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.3.8.3.2 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The

current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.4 Data Handlers

5.4.1 EJSC.XMLDataHandler

XMLDataHandler loads the requested XML file, determines the format (Full, Short or Compact) and extracts the data points into a format that the Series classes can manipulate.

The constructor expects the URL to the XML data file and an optional set of object properties.

Constructor

```
EJSC.XMLDataHandler( string url [, object options] )
```

Example

```
var myDataHandler = new EJSC.XMLDataHandler(
        "http://www.ejschart.com/data.xml"
);
```

Formats:

The XMLDataHandler supports three formats:

- Full
- Short
- Compact

5.4.1.1 Properties

5.4.1.1.1 requestType (inherited)

Definition

string **requestType** = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

5.4.1.1.2 url (inherited)

Definition

string **url** = null

Description

Defines the url currently being used to retrieve data. Descendants of AjaxDataHandler generally allow this to be set in the constructor method. To modify this value after creation, call the setUrl method which allows the data to be reloaded immediately.

5.4.1.1.3 urlData (inherited)

Definition

string **urlData** = null

Description

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

NOTE: For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"
urlData = "?param1=30"

or

url = "http://localhost/getData?param1=30"
urlData = "&extra=1"
```

5.4.1.2 Methods

5.4.1.2.1 getUrl (inherited)

Definition

string **getUrl**( )

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECIATED, see EJSC.AjaxDataHandler.url

5.4.1.2.2 loadData (inherited)

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.4.1.2.3 setRequestType (inherited)

Definition

void **setRequestType**(string requestType, boolean reload)

Description

Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.4.1.2.4 setUrl (inherited)

Definition

void **setUrl**(string url, boolean reload)

Description

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.4.1.2.5 setUrlData (inherited)

Definition

void **setUrlData**(string urlData, boolean reload)

Description

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.4.1.2.6 setXMLData (inherited)

Definition

void **setXMLData**(XMLHttpRequest request)

Description

This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the onNeedsData event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. onNeedsData was triggered), calling this method will have no effect.

5.4.1.3 Events

5.4.1.3.1 onDataAvailable (inherited)

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.4.1.3.2 onNeedsData (inherited)

Definition

boolean **onNeedsData**(EJSC.AjaxDataHandler dataHandler, EJSC.Series series, EJSC.Chart chart)
XMLHttpRequest **onNeedsData**(EJSC.AjaxDataHandler dataHandler, EJSC.Series series, EJSC.Chart chart)

Description

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

RESULT:

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its loading process.
- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

5.4.2 EJSC.XMLStringDataHandler

XMLStringDataHandler processes the provided xml-formatted string, determines the format (Full, Short or Compact) and extracts the data points into a format that the Series classes can manipulate.

The constructor expects a properly formatted string containing XML data and an optional set of object properties.

Constructor

```
EJSC.XMLStringDataHandler( string xml [, object options] )
```

Example

```
var myDataHandler = new EJSC.XMLStringDataHandler(
        '<?xml version="1.0"?><graph><plot><point x="1" y="1" /><point x="2" y="2" /><point x="
3" y="3" /></plot></graph>'
);
```

Formats:

The XMLStringDataHandler supports three formats:

- Full
- Short
- Compact

5.4.2.1 Methods

5.4.2.1.1 getXML

Definition

string **getXML**( )

Description

Returns the xml string currently stored in the XMLStringDataHandler.

5.4.2.1.2 loadData (inherited)

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.4.2.1.3 setXML

Definition

void **setXML**(string xml)

Description

Updates the xml string stored in the XMLStringDataHandler.

Note: This will not cause the data to be refreshed, see EJSC.Series.reload() for more information.

5.4.2.2 Events

5.4.2.2.1 onDataAvailable (inherited)

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.4.3 EJSC.ArrayDataHandler

ArrayDataHandler converts the JavaScript array of points into a format that the Series classes can manipulate.

The constructor expects a multi-dimensional array of data and an optional set of object properties.

Constructor

```
EJSC.ArrayDataHandler( array data [, object options] )
```

Example

```
var myDataHandler = new EJSC.ArrayDataHandler(
    [ [1,1], [2,2], [3,3], [4,4] ]
);
```

5.4.3.1 Methods

5.4.3.1.1 getArray

Definition

array **getArray**( )

Description

Returns the data array currently stored in the ArrayDataHandler.

5.4.3.1.2 loadData (inherited)

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.4.3.1.3 setArray

Definition

void **setArray**(array data)

Description

Updates the data array stored in the ArrayDataHandler.

Note: This will not cause the data to be refreshed, see EJSC.Series.reload() for more information.

5.4.3.2 Events

5.4.3.2.1 onDataAvailable (inherited)

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.4.4 EJSC.CSVFileDataHandler

CSVFileDataHandler loads the requested csv file and extracts the data points into a format that the Series classes can manipulate.

The constructor expects the URL to the csv data file and an optional set of object properties.

Constructor

```
EJSC.CSVFileDataHandler( string url [, object options] )
```

Example

```
var myDataHandler = new EJSC.CSVFileDataHandler(
    "http://www.ejschart.com/data.csv"
);
```

5.4.4.1 Properties

5.4.4.1.1 requestType (inherited)

Definition

string **requestType** = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

5.4.4.1.2 url (inherited)

Definition

string **url** = null

Description

Defines the url currently being used to retrieve data. Descendants of AjaxDataHandler generally allow this to be set in the constructor method. To modify this value after creation, call the setUrl method which allows the data to be reloaded immediately.

5.4.4.1.3 urlData (inherited)

Definition

string **urlData** = null

Description

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

NOTE: For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"
urlData = "?param1=30"

or

url = "http://localhost/getData?param1=30"
urlData = "&extra=1"
```

5.4.4.2 Methods

5.4.4.2.1 getUrl (inherited)

Definition

string **getUrl**( )

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECIATED, see EJSC.AjaxDataHandler.url

5.4.4.2.2 loadData (inherited)

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.4.4.2.3 setRequestType (inherited)

Definition

void **setRequestType**(string requestType, boolean reload)

Description

Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.4.4.2.4 setUrl (inherited)

Definition

void **setUrl**(string url, boolean reload)

Description

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.4.4.2.5 setUrlData (inherited)

Definition

void **setUrlData**(string urlData, boolean reload)

Description

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.4.4.2.6 setXMLData (inherited)

Definition

void **setXMLData**(XMLHttpRequest request)

Description

This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the onNeedsData event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. onNeedsData was triggered), calling this method will have no effect.

5.4.4.3 Events

5.4.4.3.1 onDataAvailable (inherited)

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.4.4.3.2 onNeedsData (inherited)

Definition

boolean **onNeedsData**(EJSC.AjaxDataHandler dataHandler, EJSC.Series series, EJSC.Chart chart)
XMLHttpRequest **onNeedsData**(EJSC.AjaxDataHandler dataHandler, EJSC.Series series, EJSC.Chart chart)

Description

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

RESULT:

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its loading process.
- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

5.4.5 EJSC.CSVStringDataHandler

CSVStringDataHandler converts the csv string provided into a format that the Series classes can manipulate.

The constructor expects a properly formatted string and an optional set of object properties.

Constructor

```
EJSC.CSVStringDataHandler( string data [, object options] )
```

Example

```
var myDataHandler = new EJSC.CSVStringDataHandler(
        "1|1,2|2,3|3,4|4"
);
```

5.4.5.1 Methods

5.4.5.1.1 getCSV

Definition

array **getCSV**( )

Description

Returns the csv string currently stored in the CSVStringDataHandler.

5.4.5.1.2 loadData (inherited)

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.4.5.1.3 setCSV

Definition

void **setCSV**(string csv)

Description

Updates the csv string stored in the CSVStringDataHandler.

Note: This will not cause the data to be refreshed, see EJSC.Series.reload() for more information.

5.4.5.2 Events

5.4.5.2.1 onDataAvailable (inherited)

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.5 Label Formatters

5.5.1 EJSC.NumberFormatter

Use this formatter when you want more control over the display of numeric values in your charts. It can be applied to the chart axis as well as series hints.

Constructor

EJSC.NumberFormatter([options])

Example

```
var nf = new EJSC.NumberFormatter({
        currency_symbol: "$",
        currency_position: "inner",
        negative_symbol: "()"
});
```

5.5.1.1 Properties

5.5.1.1.1 currency_align

Definition

string **currency_align** = "left"

Description

Valid options are "left" and "right".

5.5.1.1.2 currency_position

Definition

string **currency_position** = "inner"

Description

Valid options are "inner" and "outer".

5.5.1.1.3 currency_symbol

Definition

string **currency_symbol** = ""

Description

Specifies the symbol used to indicate currency.

Note: If blank "" then no currency marker is used.

5.5.1.1.4 decimal_separator

Definition

string **decimal_separator** = "."

Description

Specifies the symbol used to separate the whole number part from the fractional part.

Common values are "." or ","

5.5.1.1.5 forced_decimals

Definition

integer **forced_decimals** = undefined

Description

Specifies the number of decimal places which MUST be displayed (0's are appended if necessary).

5.5.1.1.6 negative_symbol

Definition

string **negative_symbol** = "-"

Description

Specifies the formatting of negative numbers.

Valid options are "-" and "()"

5.5.1.1.7 thousand_separator

Definition

string **thousand_separator** = ","

Description

Specifies the symbol used to separate thousands groupings.

Common values are:
"," renders as 1,000,000
"." renders as 1.000.000
"" renders as 1000000

5.5.1.1.8 variable_decimals

Definition

string **variable_decimals** = undefined

Description

Specifies the number of decimal places which MAY be displayed (values are truncated to this length)

If left undefined, values are not truncated.

5.5.1.2 Methods

5.5.1.2.1 format (inherited)

Definition

string **format**(float value, integer precision)

Description

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

5.5.2 EJSC.DateFormatter

Use this formatter when you want to display date values in your charts. It can be applied to the chart axis, series hints and cursor position labels by assigning the appropriate formatter property.

Dates are supported in UTC format by specifying a JavaScript date (i.e. number of milliseconds since midnight January 01, 1970) for a point value.

See the format_string property for information on supported date formatting options.

Constructor

EJSC.DateFormatter([options])

Example

```
var df = new EJSC.DateFormatter({
    format_string: "YYYY-MM-DD"
});
```

5.5.2.1 Properties

5.5.2.1.1 format_string

Definition

string **format_string** = ""

Description

The following format options are available:

Format	Description
YYYY	Four (4) digit year (i.e. 2007)
YY	Two (2) digit year (i.e. 07)
MMMM	Full month name (i.e. January)
MMM	Short month name (i.e. Jan)
MM	Two (2) digit month of the year with leading zeros (i.e. 01)
M	Month of the year without leading zeros
DDDD	Full day of the week (i.e. Monday)
DDD	Short day of the week (i.e. Mon)
DD	Two (2) digit day of the month with leading zeros (i.e. 04)
D	Day of the month without leading zeros
HH	Hour of the day in 24-hour format with leading zeros
H	Hour of the day in 24-hour format without leading zeros
hh	Hour of the day in 12-hour format with leading zeros
h	Hour of the day in 12-hour format without leading zeros
NN	Minutes with leading zeros
N	Minutes without leading zeros
SS	Seconds with leading zeros
S	Seconds without leading zeros
ZZZ	Milliseconds (3 places) with leading zeros
ZZ	Milliseconds (2 places) with leading zeros
Z	Milliseconds without leading zeros
AA	AM/PM notation
A	A/P notation
aa	am/pm notation
a	a/p notation

5.5.2.1.2 timezoneOffset

Definition

integer **timezoneOffset** = undefined

Description

Used in conjunction with useUTC, this property may be used to display dates in the specified time zone. The timezoneOffset property is set by specifying the number of minutes (positive or negative) to offset the given time.

Example

>> Display times in Eastern Standard Time (EST)

```
var chart = new EJSC.Chart("myChart");
    chart.x_axis_formatter = new EJSC.DateFormatter({
        format_string: "HH:NN:SS",
        useUTC: true,
        timezoneOffset: -300
    });
```

5.5.2.1.3 useUTC

Definition

boolean **useUTC** = true

Description

Set this property to false to use the workstation local time to calculate dates. This may be useful when passing in dates generated by JavaScript on the client machine to indicate current time.

5.5.2.2 Methods

5.5.2.2.1 format (inherited)

Definition

string **format**(float value, integer precision)

Description

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

5.6 Base Classes

5.6.1 EJSC.Inheritable

Base class from which all EJSC classes descend. There are no public properties, methods or events for this class at this time.

Constructor

none

5.6.2 EJSC.Series

Base series class that holds all the generic information for each series that is being created. All series should descend from this class or one of its descendants in order to function properly with the

EJSC.Chart class.

Constructor

none

5.6.2.1 Properties

5.6.2.1.1 autosort

Definition

boolean **autosort** = true

Description

Defines whether the series applies the appropriate sort to points prior to drawing. Drawing routines are generally optimized to accept data in a certain order and this property eliminates the need for the developer to worry about that order prior to sending data to the series. If set to false, the data must be properly ordered beforehand in order to ensure the series renders correctly.

Note: May not be applicable to all series (i.e. EJSC.PieSeries, EJSC.AnalogGaugeSeries)

5.6.2.1.2 color

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.6.2.1.3 coloredLegend

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.6.2.1.4 delayLoad

EXPERIMENTAL

Definition

boolean **delayLoad** = true

Description

This property allows a series to force its data handler to begin data retrieval as soon as it is added to a chart. When set to false, the series will initiate data retrieval as soon as it is added to a chart. When true, the data retrieval is initialized when the series first needs to draw.

5.6.2.1.5 legendIsVisible

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.6.2.1.6 lineOpacity

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.6.2.1.7 lineWidth

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.6.2.1.8 opacity

Definition

integer **opacity** = 50

Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setOpacity()

5.6.2.1.9 title

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.6.2.1.10 visible

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.6.2.1.11 x_axis_formatter

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.6.2.1.12 y_axis_formatter

Definition

string **y_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the Y values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.6.2.2 Methods

5.6.2.2.1 getDataHandler

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.6.2.2.2 getVisibility

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.6.2.2.3 hide

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.6.2.2.4 hideLegend

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.6.2.2.5 reload

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.6.2.2.6 setColor

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.6.2.2.7 setColoredLegend

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.6.2.2.8 setDataHandler

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.6.2.2.9 setLineOpacity

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

5.6.2.2.10 setLineWidth

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.6.2.2.11 setOpacity

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

5.6.2.2.12 setTitle

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.6.2.2.13 show

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.6.2.2.14 showLegend

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.6.2.3 Events

5.6.2.3.1 onAfterDataAvailable

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.6.2.3.2 onAfterVisibilityChange

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.6.2.3.3 onBeforeVisibilityChange

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.6.3 EJSC.GaugeSeries

Base series class that holds all the generic information for each gauge series that is being created. All gauge type series should descend from this class or one of its descendants in order to function properly with the EJSC.Chart class.

Constructor

none

5.6.3.1 Properties

5.6.3.1.1 color (inherited)

Definition

string **color** = undefined

Description

Defines the series color. If left undefined, the color is pulled from the array of default colors stored in

the chart.

Note: May not be applicable to all series (i.e. EJSC.PieSeries)

5.6.3.1.2 coloredLegend (inherited)

Definition

boolean **coloredLegend** = true

Description

Determines whether the legend text inherits the series color. Setting to false will allow the legend text to inherit its normal cascaded color. To modify this property after series creation see EJSC.Series.setColoredLegend()

5.6.3.1.3 legendIsVisible (inherited)

Definition

boolean **legendIsVisible** = true

Description

Determines whether the legend item associated with the series appears in the legend window. Use the Series.showLegend and Series.hideLegend methods to control legend item visibility after series creation.

5.6.3.1.4 lineOpacity (inherited)

Definition

integer **lineOpacity** = 100

Description

Determines the line opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setLineOpacity()

5.6.3.1.5 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.6.3.1.6 opacity (inherited)

Definition

integer **opacity** = 50

Description

Determines the fill opacity (if applicable) of the series. The range is a percentage and should be specified as an integer between 0 and 100. To modify this property after series creation see EJSC.Series.setOpacity()

5.6.3.1.7 title (inherited)

Definition

string **title** = "Series <index>"

Description

Defines the series title used in hints and legend display.

Note: The default title of Series <index> (where <index> equals the current position in the series array for its owner chart) is assigned when a series is added to a chart using addSeries. This default title is only applied if the title is blank (i.e. "") at the time the addSeries method is called.

5.6.3.1.8 visible (inherited)

Definition

boolean **visible** = true

Description

Defines whether the series is visible and can draw on the chart

5.6.3.1.9 x_axis_formatter (inherited)

Definition

string **x_axis_formatter** = undefined

Description

Defines the formatter that will be used to format hints for the X values before displaying them. If left undefined, the series will inherit the axis formatter of its parent chart.

Types:
EJSC.DateFormatter
EJSC.NumberFormatter

5.6.3.2 Methods

5.6.3.2.1 getDataHandler (inherited)

Definition

EJSC.DataHandler **getDataHandler**( )

Description

Returns the EJSC.DataHandler descendant currently assigned to the series. This is not applicable to all series and will return null if a data handler has not been defined or is not used.

5.6.3.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.6.3.2.3 hide (inherited)

Definition

void **hide**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already hidden.

5.6.3.2.4 hideLegend (inherited)

Definition

void **hideLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already hidden.

5.6.3.2.5 reload (inherited)

Definition

void **reload**( )

Description

Triggers a series reload if the functionality is defined in a child class. This may involve retrieving data again or simply recalculating. While the reload() method exists in the base Series class, implementation of the protected methods triggered by calling reload() is at the discretion of the child class.

5.6.3.2.6 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.6.3.2.7 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

Sets the EJSC.Series.coloredLegend property and updates the legend to reflect the change.

5.6.3.2.8 setDataHandler (inherited)

Definition

void **setDataHandler**(EJSC.DataHandler dataHandler, boolean reload)

Description

Updates the series data handler and triggers a data reload if reload parameter is **true**. This method may not be implemented in all child classes.

5.6.3.2.9 setLineOpacity (inherited)

Definition

void **setLineOpacity**(integer opacity)

Description

Sets the EJSC.Series.lineOpacity property and redraws the series to reflect the change.

5.6.3.2.10 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.6.3.2.11 setOpacity (inherited)

Definition

void **setOpacity**(integer opacity)

Description

Sets the EJSC.Series.opacity property and redraws the series to reflect the change.

5.6.3.2.12 setTitle (inherited)

Definition

void **setTitle**(string title)

Description

Changes the series title and update the legend caption (if applicable).

Note: this will not cause updates to a hint which is visible at the time it is called, though the next time a hint is rendered it will show the new property value.

5.6.3.2.13 show (inherited)

Definition

void **show**( )

Description

Changes the series visible state, updates the legend (if applicable) and redraws the owner chart

No effect if the series is already visible.

5.6.3.2.14 showLegend (inherited)

Definition

void **showLegend**( )

Description

Changes the series legend item visible state.

No effect if the series legend item is already visible.

5.6.3.3 Events

5.6.3.3.1 onAfterDataAvailable (inherited)

Definition

boolean **onAfterDataAvailable**(EJSC.Chart chart, EJSC.Series series)

Description

Fired after the series has processed its data and before the chart is told to redraw. Returning **false** from this event handler will cancel redrawing the chart.

5.6.3.3.2 onAfterVisibilityChange (inherited)

Definition

boolean **onAfterVisibilityChange**(EJSC.Series series, EJSC.Chart chart, boolean visible)

Description

Fired when the series visibility changes. Sends series which changed, its owner chart and its new visible property value.

Returning **false** from this event handler will cancel redrawing the chart.

5.6.3.3.3 onBeforeVisibilityChange (inherited)

Definition

boolean **onBeforeVisibilityChange**(EJSC.Series series, EJSC.Chart chart)

Description

Fired before the series visibility is about to change. Sends series which changed, its owner chart. The current visible state of the series may be checked using the series.visible property.

Returning **false** from this event handler will cancel the change in visibility.

5.6.4 EJSC.Point

Base point class that holds all the generic information for each point that is being created. All points should descend from this class or one of its descendants in order to function properly with the EJSC.Chart and EJSC.Series (and descendant) classes.

Constructor

none

5.6.4.1 Properties

5.6.4.1.1 label

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

5.6.4.1.2 userdata

Definition

string **userdata** = null

Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the EJSC.XMLDataHandler (full and short formats).

5.6.4.1.3 x

Definition

float **x** = null

Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

5.6.4.1.4 y

Definition

float **y** = null

Description

Defines point Y value.

5.6.5 EJSC.DataHandler

Base data handler class that defines properties, methods and events common to all data handler descendants. This class does not implement any actual data loading and should not be instantiated. All data handlers should descend from DataHandler or one of its descendants in order to function properly with the EJSC.Series descendants.

Constructor

none

5.6.5.1 Methods

5.6.5.1.1 loadData

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.6.5.2 Events

5.6.5.2.1 onDataAvailable

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.6.6 EJSC.AjaxDataHandler

Base data handler class for all Ajax data handlers (i.e. EJSC.XMLDataHandler, EJSC.CSVFileDataHandler). This base class defines common properties, methods and events but does not implement any data processing. This class should not be used directly.

Constructor

none

5.6.6.1 Properties

5.6.6.1.1 requestType

Definition

string **requestType** = "GET"

Description

Defines the type of HTTP request to be made. Valid options are "GET" and "POST"

5.6.6.1.2 url

Definition

string **url** = null

Description

Defines the url currently being used to retrieve data. Descendants of AjaxDataHandler generally allow this to be set in the constructor method. To modify this value after creation, call the setUrl method which allows the data to be reloaded immediately.

5.6.6.1.3 urlData

Definition

string **urlData** = null

Description

Defines the data to be included for both POST and GET requests. When performing a GET request, this data (if defined) will be appended to the URL before the request is made. For POST requests, this data will be sent as the body of the request.

NOTE: For GET requests you must include the proper syntax for appending the url stored in the url property. Example:

```
url = "http://localhost/getData"
urlData = "?param1=30"

or

url = "http://localhost/getData?param1=30"
urlData = "&extra=1"
```

5.6.6.2 Methods

5.6.6.2.1 getUrl

Definition

string **getUrl**( )

Description

Returns the url string currently stored in the AjaxDataHandler descendant.

DEPRECIATED, see EJSC.AjaxDataHandler.url

5.6.6.2.2 loadData (inherited)

Definition

void **loadData**( )

Description

Placeholder method which should be used to clear (if stored) and reload data. This method must be overridden in all descendant classes.

5.6.6.2.3 setRequestType

Definition

void **setRequestType**(string requestType, boolean reload)

Description

Updates the requestType stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Valid options for requestType are "GET" and "POST"

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.6.6.2.4 setUrl

Definition

void **setUrl**(string url, boolean reload)

Description

Updates the url string stored in the AjaxDataHandler descendant and optionally reloads the data immediately.

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.6.6.2.5 setUrlData

Definition

void **setUrlData**(string urlData, boolean reload)

Description

Updates the urlData stored in the AjaxDataHandler descendant and optionally reloads / reprocesses the data source and updates the series.

Note: If the reload parameter is omitted or false, the data will not reload automatically and must be done manually via EJSC.Series.reload()

5.6.6.2.6 setXMLData

Definition

void **setXMLData**(XMLHttpRequest request)

Description

This method allows passing of an already retrieved XMLHttpRequest object to an AjaxDataHandler descendant. When used in conjunction with the onNeedsData event, this method allows interaction with pre-existing 3rd party Ajax libraries instead of the request pool built into the chart library.

If the data handler is not currently in the loading state (i.e. onNeedsData was triggered), calling this method will have no effect.

5.6.6.3 Events

5.6.6.3.1 onDataAvailable (inherited)

Definition

void **onDataAvailable**()

Description

Placeholder for storing an event to fire after the a descendant class has loaded and processed its data.

Important: This event should be assigned by the EJSC.Series object.

5.6.6.3.2 onNeedsData

Definition

boolean **onNeedsData**(EJSC.AjaxDataHandler dataHandler, EJSC.Series series, EJSC.Chart chart)
XMLHttpRequest **onNeedsData**(EJSC.AjaxDataHandler dataHandler, EJSC.Series series, EJSC.Chart chart)

Description

This event allows for additional control over when and how data is retrieved by the data handler. If assigned, the event will be triggered whenever the its owner series has requested the data handler retrieve its data. The event passes a reference to the current data handler, its owner series and the owner chart.

The result of this event is expected to be either a boolean value or an XMLHttpRequest object which has already been retrieved (responseXML is a valid, well-formed chart xml document).

RESULT:

- **true:** The data handler will ignore other properties such as url and requestType and hold itself in a loading state and wait for its setXMLData method to be called and provided a valid XMLHttpRequest object. In this manner, a separate Ajax library may be used to retrieve data.
- **false:** The data handler will ignore other properties such as url and requestType and cancel its loading process.
- **XMLHttpRequest:** The data handler will ignore other properties such as url and requestType and immediately begin processing the data stored in responseXML.

5.6.7 EJSC.Formatter

The top level Formatter class is used as a base for all label formatters. It defines a format_string property as a guide for descendants. The format method is required to be overridden in all descendant classes in order to function properly when used with the chart classes.

Constructor

EJSC.Formatter()

5.6.7.1 Properties

5.6.7.1.1 format_string

Definition

string **format_string** = undefined

Description

Defines the format string. Classes that descend from EJSC.Formatter may use this property to store their custom format strings.

5.6.7.2 Methods

5.6.7.2.1 format

Definition

string **format**(float value, integer precision)

Description

Format is called when a value needs to be formatted for display in the axis, hint, cursor position, etc.

The behavior of the format method is to round a numeric value to the precision specified. Descendants do not need to implement the precision parameter.

5.7 Other Classes

5.7.1 EJSC.BarPoint

BarPoint is used to store an X,Y point value (and optionally user defined data). This is used in the EJSC.BarSeries class to store data for each bar to be drawn on the chart. This object is created automatically once data is made available via a EJSC.DataHandler descendant. BarPoint objects should generally not be created manually.

The constructor expects the X and Y value as well as the EJSC.BarSeries which owns the point and optionally (leave null if not used) a userdata string value.

Constructor

```
EJSC.BarPoint( float x, float y, string userdata, EJSC.BarSeries owner )
```

5.7.1.1 Properties

5.7.1.1.1 userdata (inherited)

Definition

string **userdata** = null

Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the EJSC.XMLDataHandler (full and short formats).

5.7.1.1.2 x (inherited)

Definition

float **x** = null

Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

5.7.1.1.3 y (inherited)

Definition

float **y** = null

Description

Defines point Y value.

5.7.2 EJSC.GaugePoint

GaugePoint is used to store an X point value (and optionally a label). This is used in the EJSC.GaugeSeries class to store data the gauge indicator. This object is created automatically once data is made available via a EJSC.DataHandler descendant. GaugePoint objects should generally not be created manually.

The constructor expects the X value as well as the EJSC.GaugeSeries which owns the point and optionally (leave null if not used) a label string value.

Constructor

```
EJSC.GaugePoint( float x, string label, EJSC.GaugeSeries owner )
```

5.7.2.1 Properties

5.7.2.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

5.7.2.1.2 userdata (inherited)

Definition

string **userdata** = null

Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the EJSC.XMLDataHandler (full and short formats).

5.7.2.1.3 x (inherited)

Definition

float **x** = null

Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

5.7.3 EJSC.PiePoint

PiePoint is used to store an X point value (and optionally a label value). This is used in the EJSC.PieSeries class store data for each pie piece to be drawn on the chart. This object is created automatically by once data is made available via a EJSC.DataHandler descendant. PiePoint objects should generally not be created manually.

The constructor expects the X value as well as the EJSC.PieSeries which owns the point and optionally (leave null if not used) label and userdata string values.

Constructor

```
EJSC.PiePoint( number x, string label, string userdata, EJSC.PieSeries owner )
```

5.7.3.1 Properties

5.7.3.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

5.7.3.1.2 userdata (inherited)

Definition

string **userdata** = null

Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the EJSC.XMLDataHandler (full and short formats).

5.7.3.1.3 x (inherited)

Definition

float **x** = null

Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

5.7.4 EJSC.XYPoint

XYPoint is used to store a X,Y point value. This is used in series such as EJSC.LineSeries, EJSC.AreaSeries, EJSC.ScatterSeries and EJSC.BarSeries to store each point to be drawn on the chart. This object is created automatically by each series once data is made available via a EJSC.DataHandler descendant. XYPoint objects should generally not be created manually.

The constructor expects the X and Y values as well as the EJSC.Series which owns the point and an optional (leave null if not used) userdata string value.

Constructor

```
EJSC.XYPoint( number x, number y, string userdata, EJSC.Series owner )
```

5.7.4.1 Properties

5.7.4.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

5.7.4.1.2 userdata (inherited)

Definition

string **userdata** = null

Description

Stores a user-defined string related to the point. This may be used to hold information such as database id values, drill down urls or any other related data. Currently only supported by the EJSC.XMLDataHandler (full and short formats).

5.7.4.1.3 x (inherited)

Definition

float **x** = null

Description

Defines the point X value. This may be mapped to a text label automatically by the data handler in instances where the point data is not numeric.

5.7.4.1.4 y (inherited)

Definition

float **y** = null

Description

Defines point Y value.

5.8 Using Colors

Using Colors

Colors may be specified in any of the following formats:

RGB: "rgb(<red>, <green>, <blue>)"

example: "rgb(255,0,0)"

RGBA: "rgba(<red>, <green>, <blue>, <opacity>)"

example: "rgba(255,0,0,50)"

HEX: "#<red><green><blue>"

example: "#ff0000"

5.9 Text Replacement Options

String	Replaced With
[chart_title]	The title of the chart
[series_title]	The title of the series the selected point resides in
[xaxis]	The text displayed on the X axis
[yaxis]	The text displayed on the Y axis
[x]	The preformatted X value of the point
[y]	The preformatted Y value of the point
[label]	The label property of the current PiePoint (only applicable to PieSeries hints)
[total]	The total value of the current PieSeries (only applicable to PieSeries hints)
[percent]	The numeric value representing the percent the current PiePoint occupies (only applicable to PieSeries hints, the value is piece value / total * 100 and the % sign is not part of the replacement)

5.10 Exporting To SVG

Including SVG Export Functionality

To add the SVG Export functionality to your current implementation, simply include the file immediately after the EJSChart.js

```
<head>
        <script type="text/javascript" src="/EJSChart/EJSChart.js"></script>
        <script type="text/javascript" src="/EJSChart/EJSChart_SVGExport.js"></script>
```

```
</head>
```

Exporting SVG

Exporting SVG from your chart is as simple as making a single JavaScript call. The following example shows how to call the exportSVG method and obtain the resulting SVG. The default configuration will produce complete SVG including headers and sized to match the dimensions of source chart. If additional customization is needed, the options parameter allows for settings to be changed at the time of export.

```
<script type="text/javascript">

        var chart = new EJSC.Chart("myChart", {});
        var series = new EJSC.LineSeries(
                new EJSC.ArrayDataHandler([[1,1],[2,2],[3,3],[4,2],[5,1]]),
                {}
        );
        chart.addSeries(series);

        function buttonClick() {
                alert(chart.exportSVG());
        }

</script>
<button onclick="buttonClick();">Export SVG</button>
```

6 Getting Support

There are a wide variety of technical support options available for Emprise software products. For users who have purchased the Developer or Enterprise editions, and customers with maintenance plans, support options may include Priority E-mail Support and our Technical Support Hotline. Technical support documents, help files and access to our support forums are available to all users of Emprise software products.

Documentation

Emprise JavaScript Charts documentation is available in several formats:

- Online Web Documentation (Searchable)

Downloadable Formats:

- PDF Manual
- Windows HTML Help (.chm)
- Classic Windows Help (.hlp)

Example Charts

Our website contains a wide variety of example charts to assist you in creating and customizing your own. All examples include full JavaScript and data descriptions as well as links to the relevant help files for each property used.

- Line Chart
- Area Chart
- Bar Chart
- Pie Chart
- More...

Support Forums

Share questions, suggestions, and information about your Emprise software products with other users and the Emprise support and development staff in our Support Forums.

Contact Us

If you cannot find an answer to your question from the resources listed above, send our support department a message and we will get back to you as soon as we can.

General Information & Questions
info@ejschart.com
Sales Related Inquiries
sales@ejschart.com
Technical Support
support@ejschart.com
Phone
Sales: +1 (860) 464-8555
Support: (See support contract)

Mailing Address:
PO Box 129

Ledyard, CT 06339

USA