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:450; height:330;"></div>` where desired in the body of your webpage. The style width and height properties can be adjusted as desired.

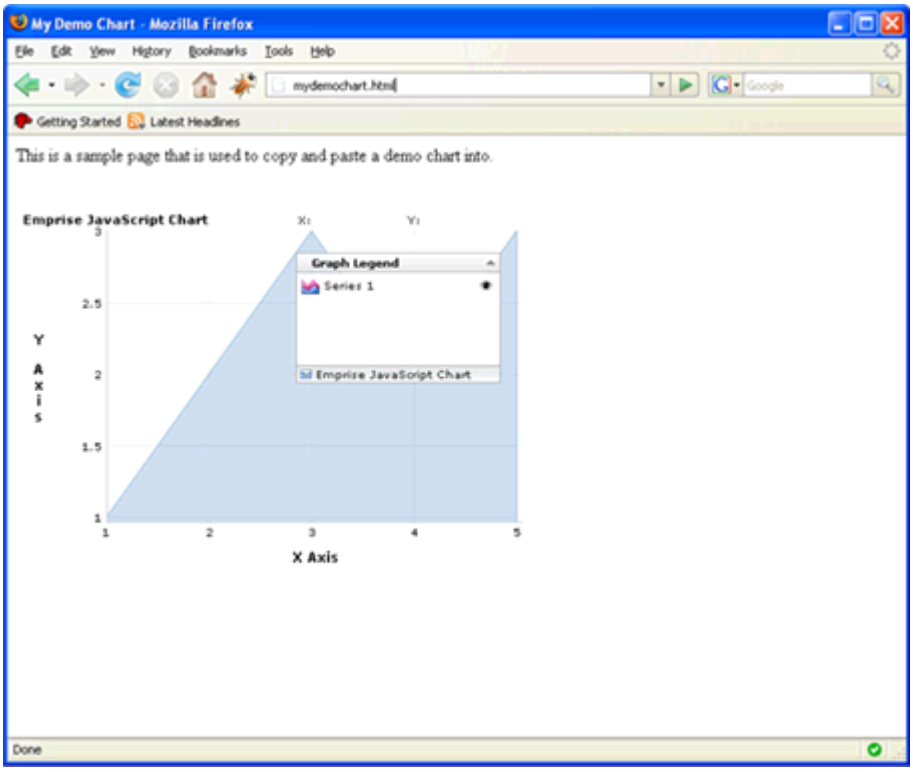
```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head>
    <title>My Demo Chart</title>
    <meta name="Author" content="Emprise Corporation">
    <meta name="Description" content="EJSChart Demo Chart HTML Page">
    <script type="text/javascript" src="/EJSChart/EJSChart.js"> </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:450; height:330;"></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>
```

```
<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:450; height:330;"></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:600; height:400;"></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:600; height:400;"></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:600; height:400;"></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:600; height:400;"></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>
```

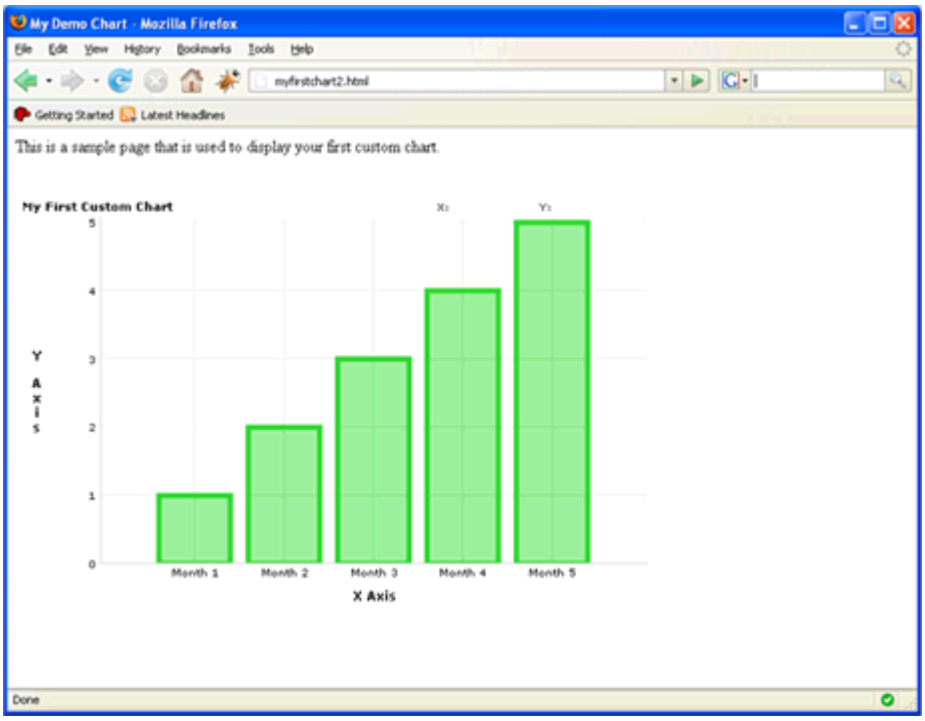
6. Once created the series must then be added to the chart using the addSeries method in the chart class.

```
<!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:600; height:400;"></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:600; height:400;"></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:

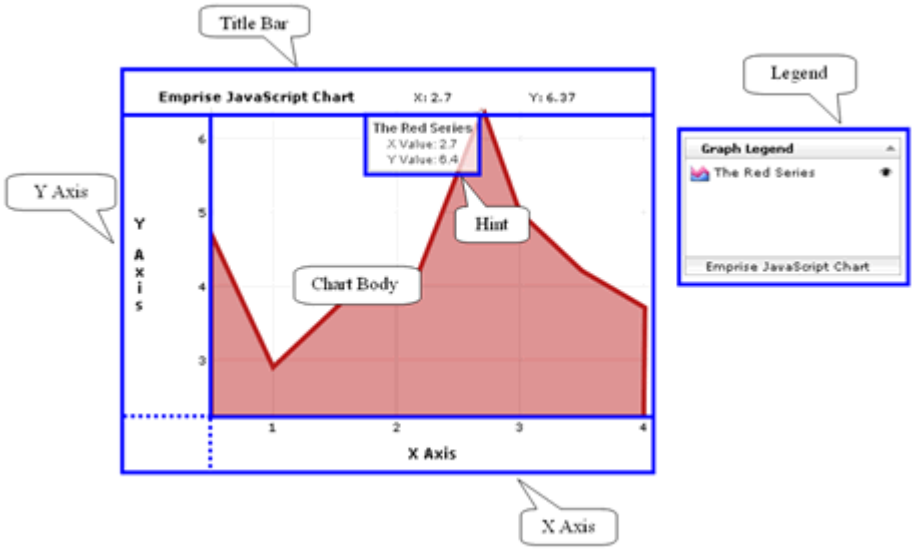
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin, {lineWidth:
4});
```

- Modification of the lineWidth property after creation:

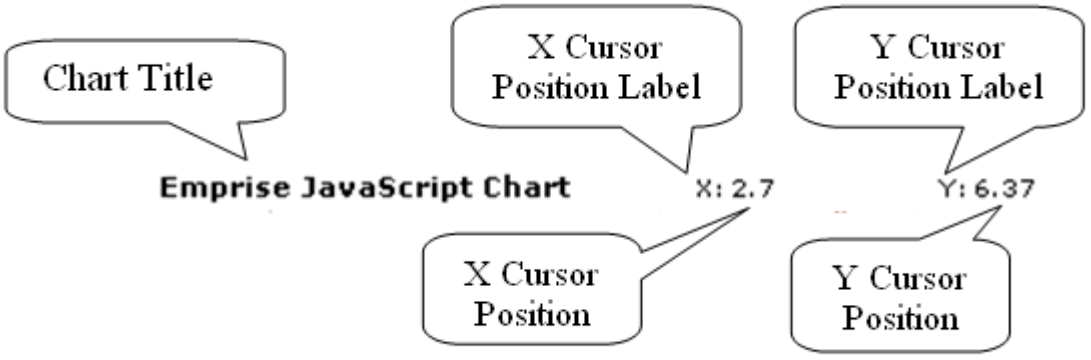
```
var myChartSeries = new EJSC.FunctionSeries(Math.sin);
    myChartSeries.setLineWidth(4);
```

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:

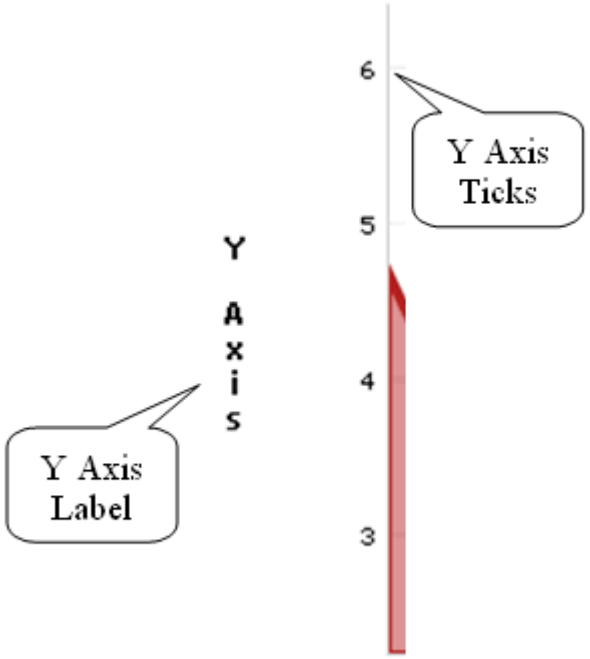
Anatomy of a Chart



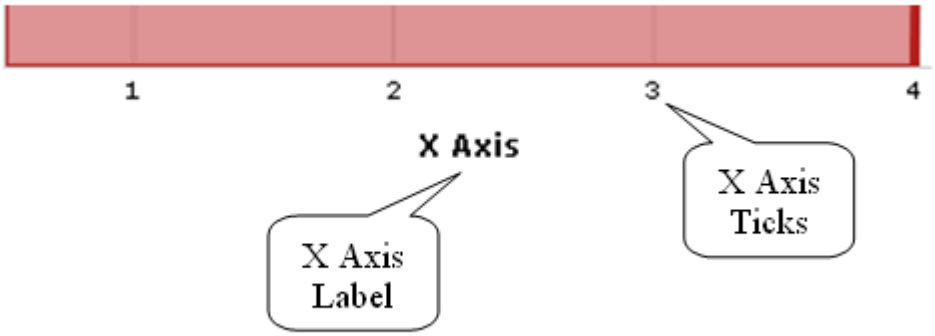
1.4.1 Titlebar



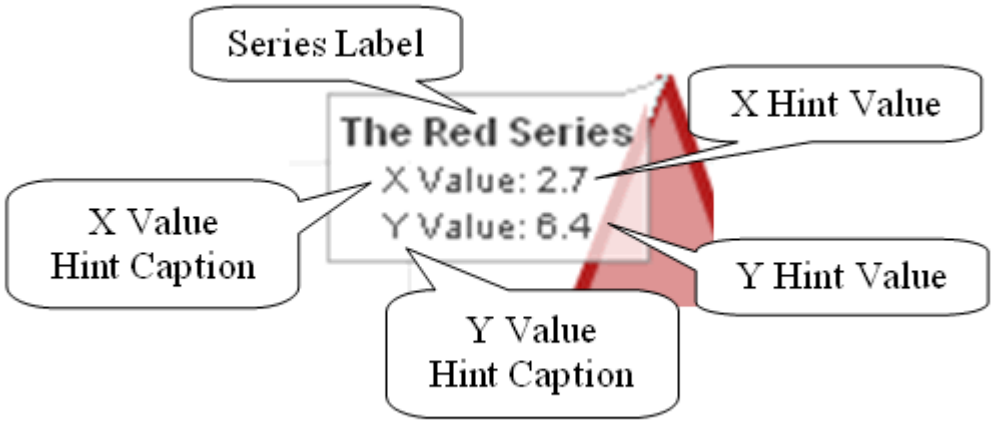
1.4.2 Y Axis



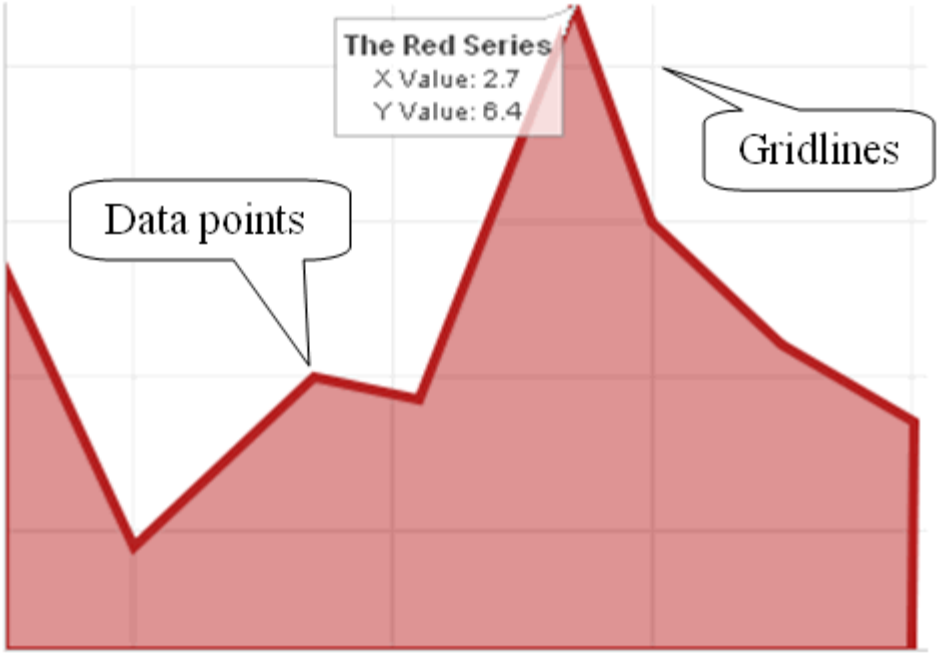
1.4.3 X Axis



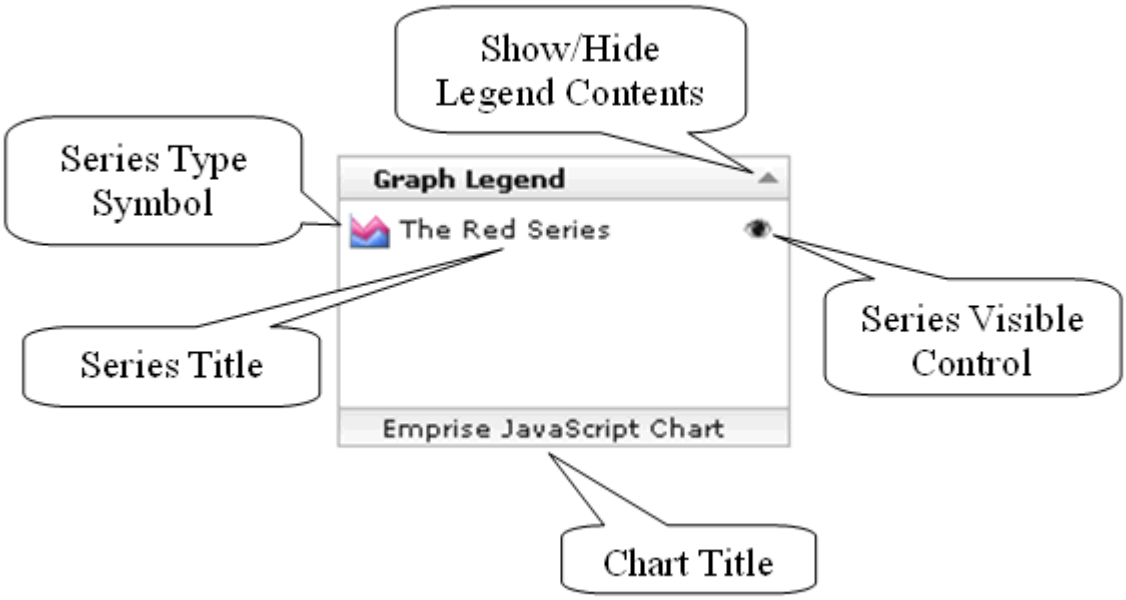
1.4.4 Hint



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 1.0

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 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.

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_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}
	);
```

5.2.1.3 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}
        );
```

5.2.1.4 auto_zoom

Definition

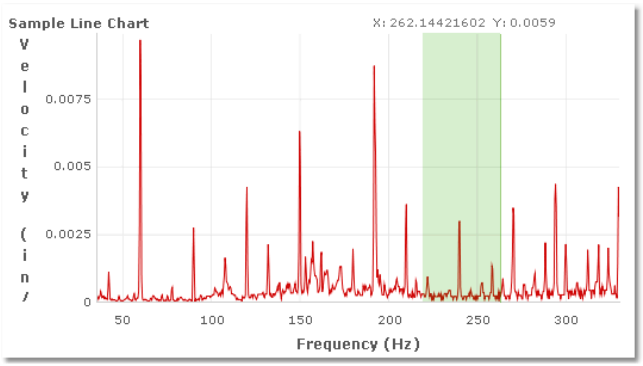
string **auto_zoom** = undefined

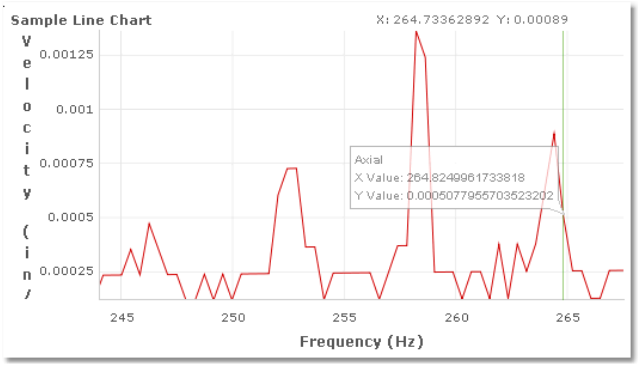
Valid property values:
'x' - **NOT IMPLEMENTED**
'y'

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" }
);
```

5.2.1.5 force_static_points

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}
        );
```

5.2.1.6 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}
        );
```

5.2.1.7 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}}
        );
```

5.2.1.8 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}
        );
```

5.2.1.9 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}
        );
```

5.2.1.10 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}
        );
```

5.2.1.11 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}
        );
```

5.2.1.12 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}
        );
```

5.2.1.13 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}
        );
```

5.2.1.14 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}
        );
```

5.2.1.15 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"}
        );
```

5.2.1.16 x_axis_caption

Definition

string **x_axis_caption** = "X Axis"

Description

Defines the text to be displayed below the X axis.

Example

>> Customize the x-axis caption.

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

5.2.1.17 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}
       );
```

5.2.1.18 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"})}
       );
```

5.2.1.19 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"}
       );
```

5.2.1.20 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}
       );
```

5.2.1.21 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}
       );
```

5.2.1.22 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:"}
        );
```

5.2.1.23 x_zero_plane

Definition

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

Description

Defines the properties of the zero plane line to be drawn at X=0. It is used to specify if the line should be shown as well as its color and thickness.

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}}
        );
```

5.2.1.24 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"}
        );
```

5.2.1.25 y_axis_extremes_ticks

NOT IMPLEMENTED
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}
       );
```

5.2.1.26 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})}
       );
```

5.2.1.27 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"}
       );
```

5.2.1.28 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}
       );
```

5.2.1.29 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}
       );
```

5.2.1.30 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:"}
       );
```

5.2.1.31 y_zero_plane

Definition

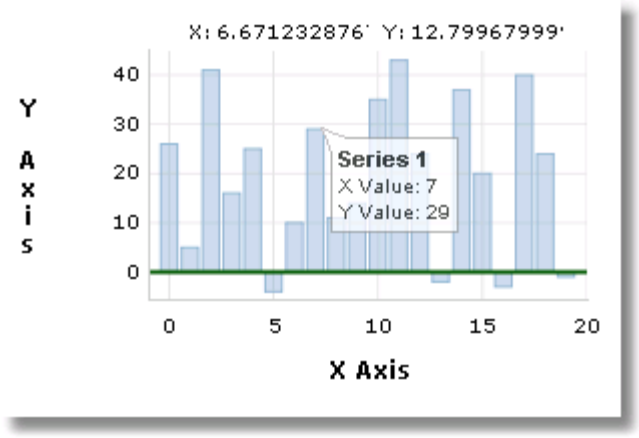
object **y_zero_plane** = {
color: "rgb(0,0,0)",
show: false,
thickness: 1
}

Description

Defines the properties of the zero plane line to be drawn at Y=0. It is used to specify if the line should be shown as well as its color and thickness.

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)

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.

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.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 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.

5.2.2.4 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.

5.2.2.5 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.

5.2.2.6 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.

5.2.2.7 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.

5.2.2.8 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

5.2.2.9 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.

5.2.2.10 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.

5.2.2.11 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 );
```

5.2.2.12 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.

5.2.2.13 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 );
```

5.2.2.14 setCrosshairs

NOT FULLY IMPLEMENTED

5.2.2.15 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.

5.2.2.16 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.

5.2.2.17 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 onAfterUnselectPoint

Definition

void **onAfterUnselectPoint**( )

Description

Called after a point has lost selection.

5.2.3.5 onAfterZoom

Definition

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

Description

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

5.2.3.6 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.

5.2.3.7 onBeforeDraw

Definition

boolean **onBeforeDraw**( )

Description

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

5.2.3.8 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.

5.2.3.9 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.

5.2.3.10 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).

5.2.3.11 onShowCrosshairs

Definition

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

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

5.2.3.12 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.

5.2.3.13 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"

5.3 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 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)

5.3.1.1.2 coloredLegend (inherited)

Definition

string **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()

5.3.1.1.3 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.1.1.4 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.

5.3.1.1.5 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.1.1.6 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

5.3.1.1.7 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 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.

5.3.1.2.5 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.1.2.6 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

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

5.3.1.2.7 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.

5.3.1.2.8 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.1.2.9 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.

5.3.1.2.10 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.

5.3.1.3 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.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" }
);
```

5.3.2.1 Properties

5.3.2.1.1 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)

5.3.2.1.2 coloredLegend (inherited)

Definition

string **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()

5.3.2.1.3 lineWidth

Definition

integer **lineWidth** = 1

Description

Defines the width of the border around bars.

5.3.2.1.4 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.

5.3.2.1.5 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.2.1.6 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

5.3.2.1.7 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

5.3.2.2 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 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.

5.3.2.2.5 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.2.2.6 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

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

5.3.2.2.7 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.

5.3.2.2.8 setLineWidth

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.2.2.9 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.

5.3.2.2.10 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.

5.3.2.3 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.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" }
);
```

5.3.3.1 Properties

5.3.3.1.1 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)

5.3.3.1.2 coloredLegend (inherited)

Definition

string **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()

5.3.3.1.3 lineWidth

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.3.1.4 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.

5.3.3.1.5 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.3.1.6 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

5.3.3.1.7 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 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.3.2.2 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.

5.3.3.2.3 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.

5.3.3.2.4 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.3.2.5 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

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

5.3.3.2.6 setLineWidth

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.3.2.7 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.

5.3.3.2.8 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.

5.3.3.3 Events

5.3.3.3.1 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.

5.3.3.3.2 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.

5.3.4 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" }
);
```

5.3.4.1 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

string **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 lineWidth

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.4.1.4 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.

5.3.4.1.5 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.4.1.6 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

5.3.4.1.7 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

Enter topic text here.

5.3.4.2.1 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.

5.3.4.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.4.2.3 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.

5.3.4.2.4 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 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.

5.3.4.2.8 setLineWidth

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.4.2.9 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.

5.3.4.2.10 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.

5.3.4.3 Events

Enter topic text here.

5.3.4.3.1 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.

5.3.4.3.2 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.

5.3.4.3.3 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.

5.3.5 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" }
);
```

5.3.5.1 Properties

5.3.5.1.1 defaultColors

Definition

array **defaultColors** = EJSC.DefaultPieColors

Description

Defines the pool of default colors available for pie pieces.

5.3.5.1.2 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.

5.3.5.1.3 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.5.1.4 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

5.3.5.2 Methods

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 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.

5.3.5.2.5 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.

5.3.5.2.6 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);
```

5.3.5.2.7 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.

5.3.5.2.8 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.

5.3.5.3 Events

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.

5.3.5.3.4 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)"

5.3.6 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" }
      );
```

5.3.6.1 Properties

5.3.6.1.1 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)

5.3.6.1.2 coloredLegend (inherited)

Definition

string **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()

5.3.6.1.3 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

5.3.6.1.4 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

5.3.6.1.5 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.

5.3.6.1.6 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.6.1.7 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

5.3.6.1.8 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

5.3.6.2 Methods

5.3.6.2.1 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.

5.3.6.2.2 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.6.2.3 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.

5.3.6.2.4 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.

5.3.6.2.5 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.6.2.6 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

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

5.3.6.2.7 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.

5.3.6.2.8 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" );
```

5.3.6.2.9 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.

5.3.6.2.10 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.

5.3.6.3 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.

5.3.7 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"

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" }
);
```

5.3.7.1 Properties

5.3.7.1.1 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)

5.3.7.1.2 coloredLegend (inherited)

Definition

string **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()

5.3.7.1.3 lineWidth (inherited)

Definition

integer **lineWidth** = 1

Description

Defines the width of the line connecting series points.

5.3.7.1.4 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.

5.3.7.1.5 visible (inherited)

Definition

boolean **visible** = true

Description

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

5.3.7.1.6 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

5.3.7.1.7 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 getVisibility (inherited)

Definition

boolean **getVisibility**( )

Description

Returns a boolean indicating the series current visible state

5.3.7.2.2 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.

5.3.7.2.3 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.

5.3.7.2.4 setColor (inherited)

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.3.7.2.5 setColoredLegend (inherited)

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

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

5.3.7.2.6 setLineWidth (inherited)

Definition

void **setLineWidth**(integer width)

Description

Updates the lineWidth property and redraws the chart.

5.3.7.2.7 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.

5.3.7.2.8 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.

5.3.7.3 Events

5.3.7.3.1 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.

5.3.7.3.2 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 Methods

5.4.1.1.1 getUrl

Definition

string **getUrl**( )

Description

Returns the url string currently stored in the XMLDataHandler.

5.4.1.1.2 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.

5.4.1.1.3 setUrl

Definition

void **setUrl**(string url)

Description

Updates the url string stored in the XMLDataHandler.

Note: This will not cause the data to be refreshed, see EJSC.Series.reload() for more information.

5.4.1.2 Events

5.4.1.2.1 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.

5.4.2 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] ]
);
```

5.4.2.1 Methods

5.4.2.1.1 getArray

Definition

array **getArray**( )

Description

Returns the data array currently stored in the ArrayDataHandler.

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 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.

5.4.2.2 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.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"
);
```

5.4.3.1 Methods

5.4.3.1.1 getUrl

Definition

string **getUrl**( )

Description

Returns the url string currently stored in the CSVFileDataHandler.

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 setUrl

Definition

void **setUrl**(string url)

Description

Updates the url string stored in the CSVFileDataHandler.

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.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"
);
```

5.4.4.1 Methods

5.4.4.1.1 getCSV

Definition

array **getCSV**( )

Description

Returns the csv string currently stored in the CSVStringDataHandler.

5.4.4.1.2 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.

5.4.4.1.3 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.

5.4.4.2 Events

5.4.4.2.1 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

5.5.2.2 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 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)

5.6.2.1.2 coloredLegend

Definition

string **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()

5.6.2.1.3 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.

5.6.2.1.4 visible

Definition

boolean **visible** = true

Description

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

5.6.2.1.5 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

5.6.2.1.6 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 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.

5.6.2.2.5 setColor

Definition

void **setColor**(string color)

Description

Changes the series color and causes the chart to redraw.

5.6.2.2.6 setColoredLegend

Definition

void **setColoredLegend**(boolean coloredLegend)

Description

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

5.6.2.2.7 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.

5.6.2.2.8 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.

5.6.2.2.9 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.

5.6.2.3 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.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

5.6.3.1 Properties

5.6.3.1.1 label

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

5.6.3.1.2 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).

5.6.3.1.3 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.

5.6.3.1.4 y

Definition

float **y** = null

Description

Defines point Y value.

5.6.4 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

5.6.4.1 Methods

5.6.4.1.1 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.

5.6.4.2 Events

5.6.4.2.1 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.

5.6.5 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()

5.6.5.1 Properties

5.6.5.1.1 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.

5.6.5.2 Methods

5.6.5.2.1 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.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 )
```

5.7.1.1 Properties

5.7.1.1.1 label (inherited)

Definition

string **label** = null

Description

Defines the point label. Currently this is only implemented in EJSC.PieSeries EJSC.PiePoint objects.

5.7.1.1.2 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).

5.7.1.1.3 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.

5.7.1.1.4 y (inherited)

Definition

float **y** = null

Description

Defines point Y value.

5.7.2 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 )
```

5.7.2.1 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.

5.8 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)

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