JS Charts User Guide

Introduction

JS Charts is a Javascript class used to draw charts using entirely client-side scripting. No additional plugins are necessary. Just include our scripts, prepare your chart data in XML or Javascript Array, and your chart is ready.

JS Charts allows you to draw different types of charts, like bars, pie charts or simple lines, and it is highly customizable.

Implementation

1. **Easy. We begin with the header of course.**

There is only one Javascript file to be included, it contains the main code and canvas functions compatible with Internet Explorer 6, 7 and 8.

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

Fig. 1 - Example of how to include the script files into the <HEAD> section of your page

Jscharts.js contains the main library for the charts and code to easily "draw" text over the graphic chart and canvas functions required by Internet Explorer (implemented natively in Firefox, Opera or Safari).

2. **Container**

The second step is to prepare the container which will hold one chart. This can be a simple *<DIV>* tag. It is mandatory for this tag to have a unique ID set.

```
<div id="chartcontainer">This is just a replacement in case Javascript is not
available or used for SEO purposes</div>
```

Fig. 2 - Container example

The container's content will be replaced by JS Charts.

3. **First chart**

Next, a three line Javascript code is needed for your first chart. First, the chart data are prepared, a simple array contains other arrays, each of two elements. Each of these two-element arrays will represent one point in a line graphic, or one bar in a bar chart, or a pie section in a pie chart. For the bar and pie chart, the array's elements are a unit name and a value, and for the line charts the elements represent two values.

These data are stored in the *myData* variable as in the following example:

```
<script type="text/javascript">
      var myData = new Array([10, 20], [15, 10], [20, 30], [25, 10], [30, 5]);
      var myChart = new JSChart('chartcontainer', 'line');
      myChart.setDataArray(myData);
      myChart.draw();
</script>
```

Fig. 3 - A simple line chart

Second line is the constructor and initializes the chart by providing the container ID, the chart type (possible values are *line*, *bar* and *pie*).

Third line introduces the data to the JSChart object.

And finally the fourth line executes the chart drawing.

```
<script type="text/javascript">
	var myData = new Array(['unit', 20], ['unit two', 10], ['unit three', 30], ['other unit', 10], ['last unit', 30]);
	var myChart = new JSChart('chartcontainer', 'bar');
	myChart.setDataArray(myData);
	myChart.draw();
</script>
```

Fig. 4 - Bar type chart; observe the data array format

```
<script type="text/javascript">
	var myChart = new JSChart('chartcontainer', 'bar');
	myChart.setDataXML('data.xml');
	myChart.draw();
</script>
```

Fig. 5 – Pie type chart using an XML file for data input

Above are two examples for bar and pie type charts. In the last example, an XML file was used to input the data intro the JSChart object. The XML file must have an exact format:

```
<?xml version="1.0"?>
<JSChart>
	<dataset type="line">
		<data unit="10" value="20"/>
		<data unit="15" value="10"/>
		<data unit="20" value="30"/>
		<data unit="25" value="10"/>
		<data unit="30" value="5"/>
	</dataset>
</JSChart>
```

Fig. 6 – XML file example

The main node must be named *JSChart*. The child nodes can be *dataset*, *colorset* and *optionset*. In the above example only the *dataset* is used, since it is the only mandatory child node. The *dataset* must define the chart type (*line* in the above example) and must contain all the chart data. The *unit* and *value* correspond to the array pair in the previous example. *Colorset* and *optionset* are described in the Customization chapter.

Multiple data series for line charts

Multiple series of data can be used in *line* type charts only. In the example below the *setDataArray* method is called twice with different arrays of data.

```
<script type="text/javascript">
	var myData = new Array([10, 20], [15, 10], [20, 30], [25, 10], [30, 5]);
	var myData2 = new Array([10, 10], [15, 5], [25, 25], [30, 20]);
	var myChart = new JSChart('chartcontainer', 'line');
	myChart.setDataArray(myData);
```

```
	myChart.setDataArray(myData2);
	myChart.draw();
</script>
```

Fig. 7 – Line chart with two series of data

The setDataArray method must be called for every line in the chart. Optionally, a second argument can be passed to the *setDataArray* method representing the name or id of the data set (this can be useful in customizing the chart – more about it in the Customization chapter).

If using setDataXML to load the data, the XML file must contain multiple *dataset* blocks like in the below example:

```
<?xml version="1.0"?>
<JSChart>
	<dataset type="line">
		<data unit="10" value="20"/>
		<data unit="15" value="10"/>
		<data unit="20" value="30"/>
		<data unit="25" value="10"/>
		<data unit="30" value="5"/>
	</dataset>
	<dataset type="line">
		<data unit="10" value="10"/>
		<data unit="15" value="5"/>
		<data unit="25" value="25"/>
		<data unit="30" value="20"/>
	</dataset>
</JSChart>
```

Fig. 8 – XML for line chart with two series of data

The number of data series which can be added in the chart is not limited. Using ids with XML datasets is also explained in the following chapters.

Customization

JS Charts is a highly customizable Javascript library. There are several API functions for you to customize your charts. There are two ways to customize the charts: using the built-in Javascript functions or using the *optionset* node in the XML file.

1. **Using the built-in functions**

A few examples of how to customize a chart:

```
<script type="text/javascript">
	var myData = new Array([10, 20], [15, 10], [20, 30], [25, 10], [30, 5]);
	var myChart = new JSChart('chartcontainer', 'line');
	myChart.setDataArray(myData);
	myChart.setBackgroundColor('#efe');
	myChart.setAxisNameX('Custom X Axis Name');
```

```
        myChart.setAxisNameY('Y Axis');
        myChart.setSize(500, 400);
        myChart.setTitle('My Chart Title');
        myChart.setTitleColor('#5555AA');
        myChart.setTitleFontSize(10);
        myChart.draw();
</script>
```

Fig. 9 – Customized chart

All public functions are described in the *Reference* chapter.

When using multiple data series in a line chart, you might want different settings for each line (e.g. different colors). This will require associating the data set with the customization method using an id. The *setDataArray* can be called using a second argument which will be the data set *id*. The same *id* must be used when calling *setLineColor*, *setLineWidth* or *setLineOpacity* to set a property for the specific data set. In the following example the *first line* id is used to colorize the first data set and the *second line* id is used to set the width of the line for the second data set. Also a tooltip is defined for the line with the *first line* id (you can read more about the tooltips in *Reference*).

```
<script type="text/javascript">
        var myData = new Array([10, 20], [15, 10], [20, 30], [25, 10], [30, 5]);
        var myData2 = new Array([10, 10], [15, 5], [25, 25], [30, 20]);
        var myChart = new JSChart('chartcontainer', 'line');
        myChart.setDataArray(myData, 'first line');
        myChart.setDataArray(myData2, 'second line');
        myChart.setBackgroundColor('#efe');
        myChart.setAxisNameX('Custom X Axis Name');
        myChart.setAxisNameY('Y Axis');
        myChart.setLineColor('#ff0f0f', 'first line');
        myChart.setLineWidth(5, 'second line');
        myChart.setSize(500, 400);
        myChart.setTitle('My Chart Title');
        myChart.setTitleColor('#5555AA');
        myChart.setTitleFontSize(10);
        myChart.setTooltip([15, 'My Tooltip', 'first line']);
        myChart.draw();
</script>
```

Fig. 10 – Associated customization methods with data sets

These *ids* do not require to be unique. The same *id* can be associated to more than one data set.

2. Using *optionset*

Optionset is a special node which can be inserted in the XML file that is used to import data into the chart.

```
<?xml version="1.0"?>
<JSChart>
	<dataset type="line">
		<data unit="10" value="20"/>
		<data unit="15" value="10"/>
		<data unit="20" value="30"/>
		<data unit="25" value="10"/>
		<data unit="30" value="5"/>
	</dataset>
	<optionset>
		<option set="setBackgroundColor" value="'#efe'"/>
		<option set="setAxisNameX" value="'Custom X Axis Name'"/>
		<option set="setAxisNameY" value="'Y Axis'"/>
		<option set="setSize" value="500,400"/>
		<option set="setTitle" value="'My Chart Title'"/>
		<option set="setTitleColor" value="'#5555AA'"/>
		<option set="setTitleFontSize" value="10"/>
	</optionset>
</JSChart>
```

Fig. 11 – Example using optionset within an XML file

The *<option>* tag has two mandatory attributes. First attribute is *set* and is actually the usual Javascript function name. Second attribute is *value* and represents the exact arguments you would use for functions, including any commas or quotes (always use single quotes within the *value* because double-quotes would alter the XML syntax).

When using multiple data sets for line charts and different customization is needed for different lines, the *datasets* must receive the *id* attribute which must be used as 2nd argument in the *options* as described in the example below:

```
<?xml version="1.0"?>
<JSChart>
	<dataset type="line" id="first line">
		<data unit="10" value="20"/>
		<data unit="15" value="10"/>
		<data unit="20" value="30"/>
		<data unit="25" value="10"/>
		<data unit="30" value="5"/>
	</dataset>
	<dataset type="line" id="second line">
		<data unit="10" value="10"/>
		<data unit="15" value="5"/>
		<data unit="25" value="25"/>
		<data unit="30" value="20"/>
```

```
	</dataset>
	<optionset>
		<option set="setBackgroundColor" value="'#efe'"/>
		<option set="setAxisNameX" value="'Custom X Axis Name'"/>
		<option set="setAxisNameY" value="'Y Axis'"/>
		<option set="setLineColor" value="'#ff0f0f','first line'"/>
		<option set="setLineWidth" value="5,'second line'"/>
		<option set="setSize" value="500,400"/>
		<option set="setTitle" value="'My Chart Title'"/>
		<option set="setTitleColor" value="'#5555AA'"/>
		<option set="setTitleFontSize" value="10"/>
		<option set="setTooltip" value="[15, 'My Tooltip', 'first line']"/>
	</optionset>
</JSChart>
```

Fig. 12 – Associated customization methods with data sets, XML style

Constructor reference

```
JSCharts(string containerid, string charttype)
```

All charts must be initialized using this constructor. *Containerid* represents the ID of the element where the chart will be generated. *Charttype* sets the type of the chart (the only possible values are line, bar and pie). *Containerid* and *charttype* are mandatory.

Chart data reference

The chart data are always represented as an array. Each array element represents a set of chart data, for example a point on a line chart. The chart data set must have a certain format depending on the chart type:

Line type – array(number|string, number)

where the first number represents the value on the horizontal axis and the second number represents the values on the vertical axis. The values on the horizontal axis must all be either numbers or strings.

Bar type – array(string, number)

where the string is the bar name on horizontal axis and the number is its value on the vertical axis.

Pie type – array(string, number)

where the string is the section name and the number represents the section value.

Line type example:

```
var myData = new Array([10, 20], [15, 10], [20, 30], [25, 10], [30, 5]);
```

Line, bar or pie type example:

```
var myData = new Array(['unit', 20], ['unit two', 10], ['unit three', 30],
['other unit', 10], ['last unit', 30]);
```

If you add the chart data by XML, the chart type has no significance on how you represent the data sets, even though you still need to mention the chart type in the *dataset* node. All data are added like in the following example:

```
<?xml version="1.0"?>
<JSChart>
      <dataset type="line">
            <data unit="10" value="20"/>
            <data unit="15" value="10"/>
            <data unit="20" value="30"/>
            <data unit="25" value="10"/>
            <data unit="30" value="5"/>
      </dataset>
</JSChart>
```

Methods reference

colorizeBars(array colordata)

Assign colors to different bars in a bar type chart. *Colordata* is an array consisting of colors expressed in hexadecimal format and must have the same length as the data length. Each color is assigned to one bar data.

Please note that the *colorizeBars()* function has no effect on line or pie charts.

The *setBarColor()* function can be used to set all bars within a bar chart to the same color; *colorize()* overrides *setBarColor()*.

```
var myColors = new Array('#0f0', '#ff0000', '#00f', '#ff0', '#00ffff');
myChart.colorizeBars(myColors);
```

The XML file can hold the colors in the *colorset* section:

```
<?xml version="1.0"?>
<JSChart>
      <dataset type="bar">
            <data unit="10" value="20"/>
            <data unit="15" value="10"/>
            <data unit="20" value="30"/>
            <data unit="25" value="10"/>
            <data unit="30" value="5"/>
      </dataset>
      <colorset>
            <color value="#0f0"/>
            <color value="#ff0000"/>
            <color value="#00f"/>
            <color value="#ff0"/>
            <color value="#00ffff"/>
      </colorset>
</JSChart>
```

The *colorset* section in the XML file can also be used to colorize a pie chart.

colorizePie(array colordata)

Similar to *colorizeBars()*, *colorizePie()* assigns colors to different pie sections in a pie type chart. *Colordata* is an array consisting of colors expressed in hexadecimal format and must have the same length as the data length. Each color is assigned to one pie data.

Please note that *colorizePie()* function has no effect on line or bar charts.

If the *colorizePie()* function is not used, the pie sections within a pie chart will have random colors.

The colorizePie() function is used very similar to the colorizeBars() function:

```
var myColors = new Array('#0f0', '#ff0000', '#00f', '#ff0', '#00ffff');

myChart.colorizePie(myColors);
```

The XML usage to colorize pie charts is identical to the method described at the *colorizeBars()* function (using the *colorset* section).

draw()

This function draws the chart within the set container with the set options and data. The data loading function has to be executed before *draw()* and any options (including colors) must be set before *draw()*.

resize(integer x, integer y)

Resize an already drawn chart. *X* and *y* must be integers representing the new width and height respectively. The default size dimension for the chart is 400px/300px.

setAxisColor(string hexcolor)

Set both axes color. Please note that this function has no effect on pie charts. Default color is *#B5B5B5*.

setAxisNameColor(string hexcolor)

Set the color for both axes names. This function has no effect on pie charts. Default color is *#999*.

setAxisNameFontSize(integer fontsize)

Set the font size for the axes names. This function has no effect on pie charts. Default font size is *11*.

setAxisNameX(string axisname)

Set name for X axis. This function has no effect on pie charts. Default name is *X*.

setAxisNameY(string axisname)

Set name for Y axis. This function has no effect on pie charts. Default name is *Y*.

setAxisPaddingBottom(integer padding)

Set the bottom padding distance between X axis and bottom *<canvas>* margin. Default is *30*.

setAxisPaddingLeft(integer padding)

Set the left padding distance between Y axis and left *<canvas>* margin. Default is *40*.

setAxisPaddingRight(integer padding)

Set the right padding distance between graph and right *<canvas>* margin. Default is *30*.

setAxisPaddingTop(integer padding)

Set the top padding distance between graph and top *<canvas>* margin. Default is *50*.

setAxisValuesColor(string hexcolor)

Set the color for both X and Y axis value. This function has no effect on pie charts. Default color is *#777*.

setAxisValuesDecimals(integer decimals)

Set number of decimals for both X and Y axis values for line charts, and for Y axis values for bar charts. This function has no effect on pie charts. Default is *auto*.

setAxisValuesFontSize(integer fontsize)

Set the font size for the values on both axes. This function has no effect on pie charts. Default color is *8*.

setAxisValuesNumberX(integer number)

Set the number of values to show on X axis. If this number is too high for all the values to fit on the axis, it will be automatically reduced. Values of *0* and *1* can be used to show all values corresponding to the input data length. This function has no effect on pie charts or bar charts. Default is *0* (auto; input data length).

setAxisValuesNumberY(integer number)

Set the number of values to show on Y axis. If this number is too high for all the values to fit on the axis, it will be automatically reduced. Values of *0* and *1* can be used to show all values corresponding to the input data length. This function has no effect on pie charts. Default is *0* (auto; input data length).

setAxisValuesPrefixX(string prefix)

Set prefix for values on X axis. This function has no effect on pie or bar charts. Default is *false*.

setAxisValuesPrefixY(string prefix)

Set prefix for values on Y axis. This function has no effect on pie charts. Default is *false*.

setAxisValuesSuffixX(string suffix)

Set suffix for values on X axis. This function has no effect on pie or bar charts. Default is *false*.

setAxisValuesSuffixY(string suffix)

Set suffix for values on Y axis. This function has no effect on pie charts. Default is *false*.

setAxisWidth(integer width)

Set axis width. This function has no effect on pie charts. Default is *2*.

setBackgroundColor(string hexcolor)

Set a background color for the whole chart. Default is *#FFF*.

setBackgroundImage(string filename)

Set a background image for the whole chart. The image will repeat itself on both axes. Default is *false*.

setBarBorderColor(string hexcolor)

Set a color for the bar borders. This function has an effect only for bar charts. Default is *#C4C4C4*.

setBarBorderWidth(integer width)

Set bars border width. This function has an effect only for bar charts. Default is *1*.

setBarColor(string hexcolor)

Set a color for all bars in a bar chart. This function is overwritten by the *colorize()* function. This function has an effect only for bar charts. Default is *#3E90C9*.

setBarOpacity(float opacity)

Set the opacity for bars. It can be any floating number between 0 and 1 (for example 0.5 will make the bars half transparent). This function has an effect only for bar charts. Default is *0.9*.

setBarSpacingRatio(integer ratio)

Set a spacing ratio between bars on bar charts. Use this to control the bars width; setting it to 0 will leave no space between the bars. Use any integer between 0 and 100. This function has an effect only for bar charts. Default is *10*.

setBarValues(boolean values)

Set this to true or false to either show or not the values on top of the bars. This function has an effect only for bar charts. Default is *true*.

setBarValuesColor(string hexcolor)

Set color for the bar's value. This function has an effect only for bar charts. Default is *#2F6D99*.

setBarValuesDecimals(integer decimals)

Set number of decimals for the bar's value. This function has an effect only on bar

charts. Default is *auto*.

setBarValuesFontSize(integer fontsize)

Set the font size for the bar's value. This function has an effect only on bar charts. Default font size is *8*.

setBarValuesPrefix(string prefix)

Set prefix for bar's values. This function has an effect only on bar charts. Default is *false*.

setBarValuesSuffix(string suffix)

Set suffix for bar's values. This function has an effect only on bar charts. Default is *false*.

setCanvasIdPrefix(string prefix)

Customize the prefix attached to the generated *<canvas>* ID to avoid any conflicts with other element IDs on your page. Default is *JSChart_*.

setDataArray(array data, string id)

Import chart data as array. See the *Implementation* chapter. The id is optional and can be used to associate the data set with a setting with the same id.

setDataXML(string filename)

Import chart data as XML. See the *Implementation* and *Customization* chapters.

setFlagColor(string hexcolor)

Set a color for the tooltip flags. Default is *#F00*.

setFlagOffset(integer offset)

Set the tooltip flag offset for pie charts. Positive numbers will place the flag outside the pie and negative numbers will place the flag inside the pie. A zero value will place the flag on the pie border. This function has an effect only on pie charts. Default is *-50*.

setFlagOpacity(float opacity)

Set the opacity for tooltip flags. It can be any floating number between 0 and 1 (for example 0.5 will make the flag half transparent). Default is *1* (opaque).

setFlagRadius(integer radius)

Set tooltip flag radius. Default is *3*.

setFlagWidth(integer width)

Set the tooltip flag border width. Default is *1*.

setGraphExtend(boolean extend)

Toggle the graph extending option. When true, the graph axes and the grid lines (if grid enabled) will extend with a length equal to the 15th part of the axis length. This can help aesthetically improve the graph. Default value is *false*.

setGraphLabel(string label)

Set string for custom graph label (watermark). Default is an empty string.

setGraphLabelColor(string hexcolor)

Set color for custom graph label. Default is *#55F*.

setGraphLabelFontSize(integer fontsize)

Set the font size for custom graph label. Default font size is *8*.

setGraphLabelOpacity(float opacity)

Set the opacity for custom graph label. It can be any floating number between 0 and 1 (for example 0.5 will make the label half transparent). Default is *0.8*.

setGraphLabelPosition(string position)

Set the custom graph label position. Possible values are *nw*, *ne*, *sw* and *se* (meaning north-west, north-east, south-west, south-east and they correspond to top-left, top-right, bottom-left and bottom-right respectively). Default value is *ne*.

setGraphLabelShadowColor(string hexcolor)

Set color for custom graph label shadow. Default is *#FFF*.

setGrid(boolean grid)

Set this to true or false to either show or not the grid behind the charts. This function has no effect on pie charts. Default is *true*.

setGridColor(string hexcolor)

Set the grid color. This function has no effect on pie charts. Default is *#C6C6C6*.

setGridOpacity(float opacity)

Set grid opacity. It can be any floating number between 0 and 1 (for example 0.5 will make the grid half transparent). This function has no effect on pie charts. Default is *0.5*.

setIntervalEndX(integer end)

Used to only show a fragment of the graph with the aid of interval settings on axes. End value of the interval on X axis must be greater than any start value if set. This function has an effect only on line charts. Default is *false*.

setIntervalEndY(integer end)

Used to only show a fragment of the graph with the aid of interval settings on axes. End

value of the interval on Y axis must be greater than any start value if set. This function has no effect on pie charts. Default is *false*.

setIntervalStartX(integer start)

Used to only show a fragment of the graph with the aid of interval settings on axes. Start value of the interval on X axis must be smaller than any end value if set. This function has an effect only on line charts. Default is *false*.

setIntervalStartY(integer start)

Used to only show a fragment of the graph with the aid of interval settings on axes. Start value of the interval on Y axis must be smaller than any end value if set. This function has no effect on pie charts. Default is *false*.

setLabelX(array label)

Define a label on the X axis. The argument is an array of two elements. The first element is the value to which the label is attached and the second element is the actual string which will show as label.

```
myChart.setLabelX([15, 'My label']);
```

In the above example, a label is defined for a line chart. The label will appear for the 15 X value of the graph.

Next example is applicable for a pie chart or a bar chart:

```
myChart.setLabelX(['unit two', 'My label']);
```

Please note that this function has no effect on pie charts.

setLabelY(array label)

Define a label on the Y axis. The argument is an array of two elements. The first element is the value to which the label is attached and the second element is the actual string which will show as label.

```
myChart.setLabelY([20, 'My label']);
```

In the above example, a label is defined for a line chart. The label will appear for the 20 Y value of the graph.

Next example is applicable for a pie chart or a bar chart:

```
myChart.setLabelY(['unit two', 'My label']);
```

Please note that this function has no effect on pie charts.

setLineColor(string hexcolor, string id)

Set the line color for line charts. This function has an effect only on line charts. Default is #3E90C9. The id is optional and can be used to associate the setting with a data set with the same id.

setLineOpacity(float opacity, string id)

Set lines opacity for line charts. It can be any floating number between 0 and 1 (for example 0.5 will make the line half transparent). This function has an effect only on line charts. Default is *0.9*. The id is optional and can be used to associate the setting with a data set with the same id.

setLineWidth(integer width, string id)

Set lines width for line charts. This function has an effect only on line charts. Default is *2*. The id is optional and can be used to associate the setting with a data set with the same id.

setPieOpacity(float opacity)

Set pie opacity for pie charts. It can be any floating number between 0 and 1 (for example 0.5 will make the pie half transparent). This function has an effect only on pie charts. Default is *1* (opaque).

setPiePosition(integer x, integer y)

Set the pie position within the *<canvas>*. This function has an effect only on pie charts. A *0* value on X or Y will center the pie on that axis. Default values are both *0*.

setPieRadius(integer radius)

Set the pie radius. This function has an effect only on pie charts. Default value is the shortest *<canvas>* side divided by 3.75.

setPieUnitsColor(string hexcolor)

Set the color for the pie unit texts (section names). This function has an effect only on pie charts. Default is *#777*.

setPieUnitsFontSize(integer fontsize)

Set the unit texts font size. This function has an effect only on pie charts. Default is *8*.

setPieUnitsOffset(integer offset)

Set the unit texts offset for pie charts. Positive numbers will place the text outside the pie and negative numbers will place the text inside the pie. This function has an effect only on pie charts. Default is *10*.

setPieValuesColor(string hexcolor)

Set the value texts color. This function has an effect only on pie charts. Default is *#fff*.

setPieValuesDecimals(integer decimals)

Set number of decimals for values on pie. This function has an effect only on pie charts. Default is *auto*.

setPieValuesFontSize(integer fontsize)

Set font size for value texts on pie charts. This function has an effect only on pie charts. Default is *8*.

setPieValuesOffset(integer offset)

Set the value texts offset for pie charts. Positive numbers will place the text outside the pie and negative numbers will place the text inside the pie. This function has an effect only on pie charts. Default is *-20*.

setPieValuesPrefix(string prefix)

Set prefix for value texts on pie charts. This function has an effect only on pie charts. Default is *false*.

setPieValuesSuffix(string suffix)

Set suffix for value texts on pie charts. This function has an effect only on pie charts. Default is *false*.

setShowXValues(boolean show)

Toggle either to show or not the values on X axis or section names on pie charts. This function has no effect on pie charts. Default is *true*.

setShowYValues(boolean show)

Toggle either to show or not the values on Y axis or values on pie charts. This function has no effect on pie charts. Default is *true*.

setSize(integer x, integer y)

Predefine the *<canvas>* size, before drawing the chart. Default sizes are *400*/*300*.

setTextPaddingBottom(integer padding)

Set the bottom padding distance between X axis text and bottom *<canvas>* margin. Default is *1*.

setTextPaddingLeft(integer padding)

Set the left padding distance between Y axis text and left *<canvas>* margin. Default is *8*.

setTextPaddingTop(integer padding)

Set the top padding distance between graph title and top *<canvas>* margin. Default is *15*.

setTitle(string title)

Set the chart title. If you want no title to show, set this to an empty string. Default is *JSChart*.

setTitleColor(string hexcolor)

Set the chart title color. Default is *#8E8E8E*.

setTitleFontSize(integer fontsize)

Set font size for the chart title. Default is *11*.

setTitlePosition(string position)

Set the title position horizontally. Possible values are *center*, *left* and *right*. Left and right position are influenced by the *setAxisPaddingLeft()* and *setAxisPaddingRight()* functions since the title will be aligned with the left/right margins of the chart. Default value is *center*.

setTooltip(array tooltip, function callback)

Define a tooltip. The first argument is an array of one, two or three elements. The first element is the value to which the tooltip is attached (must be one of the input values), and the second element is an optional string which will show in the tooltip. By default, the tooltip contents are the attached values and are replaced by the optional string.

```
myChart.setTooltip([15]);

myChart.setTooltip([30, 'custom<br>contents']);
```

In the above example, two tooltips are defined for a line chart. The first will appear for the 15 value on the X axis of the graph. The second will appear for the 30 value on the X axis with a custom content.

For line charts, a third optional element can be used in the array to set the tooltip for the line with the same id only, useful if using multiple lines charts. Only works with line charts.

```
myChart.setTooltip([15, '', 'first line']);

myChart.setTooltip([30, 'custom<br>contents', 'second line']);
```

Next example is applicable for a pie chart or a bar chart:

```
myChart.setTooltip(['unit two']);

myChart.setTooltip(['last unit', 'custom<br>contents']);
```

The second argument is optional and must be a function to be called when the user clicks on the tooltip flag:

```
myChart.setTooltip([15], function(){alert('callback');});
```

If instead the tooltip contents string, the boolean false is used, no tooltip will be displayed but the tooltip flag cycle will show and any callback function will also work:

```
myChart.setTooltip([15, false], callback);
```

setTooltipBackground(string hexcolor)

Set the tooltips background color. Default value is *#e6e6e6*.

setTooltipBorder(string css)

Set the tooltips border styling. It must be a CSS expression. Default is *1px solid #d3d3d3*.

setTooltipFontColor(string hexcolor)

Set the tooltip's content color. Default is *#00F*.

setTooltipFontFamily(string font)

Set the tooltips font family. You can specify any font family like in a CSS declaration. Default value is *arial*.

setTooltipFontSize(integer fontsize)

Set the tooltips contents font size. Default is *12*.

setTooltipOffset(integer offset)

Set the tooltip distance from the tooltip flag to the tooltip itself. Default is *7*.

setTooltipOpacity(float opacity)

Set tooltips opacity. It can be any floating number between 0 and 1 (for example 0.5 will make the tooltips half transparent, 0 transparent and 1 opaque). Default is *0.7*.

setTooltipPadding(string css)

Set the tooltips padding style. It must be a CSS expression. Default is *2px 5px*.

setTooltipPosition(string position)

Set the tooltip position from the tooltip flag. Possible values are *nw*, *ne*, *sw* and *se* (meaning north-west, north-east, south-west, south-east and they correspond to top-left, top-right, bottom-left and bottom-right respectively). Default value is *se*.