# Säulendiagramm-Dokumentation – Sporenberg/08.01.2013

## *Beispiel*

```
<script>
   window.onload = function ()
   {   // Some data that is to be shown on the bar chart. To show a stacked or grouped chart
       // each number should be an array of further numbers instead.
       var data = [280,45,133];

       // Create the br chart. The arguments are the ID of the canvas tag and the data
       var bar = new RGraph.Bar('myCanvas', data);

       // Now configure the chart to appear as wanted by using the .Set() method.
       // All available properties are listed below.
       bar.Set('chart.labels', ['Richard', 'Alex', 'Nick']);
       bar.Set('chart.gutter.left', 45);
       bar.Set('chart.background.barcolor1', 'white');
       bar.Set('chart.background.barcolor2', 'white');
       bar.Set('chart.background.grid', true);
       bar.Set('chart.colors', ['red']);

       // Now call the .Draw() method to draw the chart
       bar.Draw();
   }
</script>
```

## *Properties*

You can use these properties to control how the bar chart apears. You can set them by using the Set() method. Eg:

**myBar.Set('name', 'value');**

**Background**
**Axes properties**
**Colors**
**Margins**
**Labels and text**
**Titles**
**Shadow**
**Scale**
**Key**
**Interactive features**
**Zoom**
**Events**
**Miscellaneous**

## *Background*

**chart.background.barcolor1**
The color of the background bars, (1 of 2).
*Default: white*

**chart.background.barcolor2**
The color of the background bars, (2 of 2).
*Default: white*

**chart.background.grid**
Whether to show the background grid or not.
*Default: true*

**chart.background.grid.color**
The color of the background grid.
*Default: #ddd*

**chart.background.grid.hsize**
The horizontal background grid size.
*Default: 40*

**chart.background.grid.vsize**
The vertical background grid size.
*Default: 18*

**chart.background.grid.width**
The width that the background grid lines are. Decimals (eg 0.5) are permitted.
*Default: 1*

**chart.background.grid.border**
Determines whether a border line is drawn around the grid.
*Default: true*

**chart.background.grid.hlines**
Determines whether to draw the horizontal grid lines.
*Default: true*

**chart.background.grid.vlines**
Determines whether to draw the vertical grid lines.
*Default: true*

**chart.background.grid.autofit**
Instead of specifying a pixel width/height for the background grid, you can use autofit and specify how many horizontal and vertical lines you want.
*Default: true*

**chart.background.grid.autofit.numhlines**
When using autofit this allows you to specify how many horizontal grid lines you want.
*Default: 5*

**chart.background.grid.autofit.numvlines**
When using autofit this allows you to specify how many vertical grid lines you want.
*Default: 20*

**chart.background.grid.autofit.align**
If you want to have your grid lines line up with the labels (both X and Y axes), you can set this to true and RGraph will attempt to make the grid lines line up. If you have a *chart.hmargin* set then the alignment will be thrown out.
*Default: false*

**chart.background.hbars**
An array of information stipulating horizontal coloured bars. You can use these to indicate limits. Eg: *myBar.Set('hbars', [[75, 10, 'yellow'], [85, 15, 'red']]);* This would give you two bars, one red and a lower yellow bar. The units correspond to your scale, and are the starting point and the height.
*Default: null*

**chart.background.image**
If you want to specify a background image to use on your chart, specify it with this property. If you use effects with a background image on your chart it make the effect flicker.
*Default: null*

**chart.background.image.stretch**
By default your background image is stretched (if necessary) to cover the whole chart area (gutters not included). If this is not what you want then set this property to false.
*Default: true*

**chart.background.image.x**
The X position of the image. The coordinates are the top left corner of the image.
*Default: null*

**chart.background.image.y**
The Y position of the image. The coordinates are the top left corner of the image.
*Default: null*

**chart.background.image.w**
The width of the image. If you have a large canvas with many charts - you may need to specify this.
*Default: null*

**chart.background.image.h**
The height of the image. If you have a large canvas with many charts - you may need to specify this.
*Default: null*

**chart.background.image.align**
Instead of specifying the coordinates of the image, you can instead simply align it top, bottom, left or right. Examples are:

- top left

- bottom right
- bottom
- right

*Default: null*

## *Axes properties*

**chart.noaxes**
Whether the axes are drawn
*Default: false (the axes ARE drawn)*

**chart.noxaxis**
Whether the X axis is drawn
*Default: false (the X axis IS drawn)*

**chart.noyaxis**
Whether the Y axis is drawn
*Default: false (the Y axis IS drawn)*

**chart.noendxtick**
When you're combining the Bar and Line charts, you may want to use this property to stop the end X tick from being drawn.
*Default: false (the end tick IS drawn)*

**chart.numxticks**
This controls the number of X tickmarks on the X axis.
*Default: null*

## *Colors*

**chart.strokestyle**
The color of the outline of the bars.
*Default: #666*

**chart.colors**
An array of the colors of the actual bars.
*Default: An array - ['rgb(0,0,255)', '#0f0', '#00f', '#ff0', '#0ff', '#0f0']*

**chart.colors.sequential**
If true, for regular bar charts, (not stacked or grouped), the colors that you specify will be used in a sequential fashion.
*Default: false*

**chart.colors.reverse**
If true, for stacked bar charts only, the colors that you specify will be used in a reverse order to what they are normally.

*Default: false*

## *Margins*

**chart.hmargin**
The horizontal margin (in pixels) of the chart. The horizontal margin is on the inside of the axes.
*Default: 5*

**chart.gutter.left**
The left gutter of the chart, (the gutter is where the labels and title are)).
*Default: 25*

**chart.gutter.right**
The right gutter of the chart, (the gutter is where the labels and title are).
*Default: 25*

**chart.gutter.top**
The top gutter of the chart, (the gutter is where the labels and title are).
*Default: 25*

**chart.gutter.bottom**
The bottom gutter of the chart, (the gutter is where the labels and title are).
*Default: 25*

## *Labels and text*

**chart.text.color**
The color of the labels.
*Default: black*

**chart.text.size**
The size (in points) of the labels.
*Default: 10*

**chart.text.angle**
The angle of the horizontal text labels (at the bottom of the chart). This can be one of three values - 0, 45 or 90.
*Default: 0 (Horizontal)*

**chart.text.font**
The font used to render the text.
*Default: Verdana*

**chart.labels**
An array of the labels to be used on the chart.
*Default: An empty array*

**chart.labels.above**
If true, values will be shown above the bars. For regular and stacked bar charts units are included, whereas for grouped bar charts they're not (usually there isn't enough space for them).
*Default: false*

**chart.labels.above.decimals**
This stipulates how many decimals are used in the above bar labels.
*Default: 0*

**chart.labels.above.size**
The font size of the above bar labels. Useful if you only have a few bars.
*Default: false*

**chart.labels.above.angle**
You can use this to angle the text shown above the bars. It can be anything from -90 to 90 (degrees).
*Default: null*

**chart.labels.ingraph**
An array of labels for the chart which are drawn "inside" the chart. If you have 5 data points then this should have a corresponding number of elements.
*Default: null*

**chart.ylabels**
Can be *true* or *false* and determines whether the chart has Y axis labels.
*Default: true*

**chart.ylabels.count**
A value that controls how many Y labels there are. Previously (prior to 8th August 2010) this could be 1,3,5. Now it can be any number, but keep in mind that if you use this it may result in decimals.
*Default: 5*

**chart.ylabels.specific**
You can use this option to give your own Y labels (eg ['Low', 'Medium', 'High'].
*Default: null*

**chart.xlabels.offset**
This allows you finer grained control over the X label positioning if you need it.
*Default: 0*

**chart.numyticks**
The number of Y tickmarks. If you have changed the number of Y labels, you may also want to change this to match.
*Default: 10*

## *Titles*

**chart.title**
The title of the chart, if any.
*Default: null*

**chart.title.font**
The font that the title is rendered in. If not specified the chart.text.font setting is used (usually Verdana)
*Default: null*

**chart.title.size**
The size of the title. If not specified the size is usually 2pt bigger than the chart.text.size setting.
*Default: null*

**chart.title.bold**
Whather the title is bold or not.
*Default: true*

**chart.title.background**
The background color (if any) for the title.
*Default: null*

**chart.title.color**
The color of the title.
*Default: black*

**chart.title.hpos**
This allows you to completely override the horizontal positioning of the title. It should be a number between 0 and 1, and is multiplied with the whole width of the canvas and then used as the horizontal position.
*Default: null*

**chart.title.vpos**
This allows you to completely override the vertical positioning of the title. It should be a number between 0 and 1, and is multiplied with the gutter and then used as the vertical position. It can be useful if you need to have a large gutter.
*Default: null*

**chart.title.xaxis**
This allows to specify a title for the X axis.
*Default: none*

**chart.title.xaxis.size**
This allows you to specify a size for the X axis title.
*Default: null*

**chart.title.xaxis.font**
This allows to specify a font for the X axis title.
*Default: null*

**chart.title.xaxis.bold**
This controls whether the X axis title is bold or not.
*Default: true*

**chart.title.yaxis**
This allows to specify a title for the Y axis.
*Default: none*

**chart.title.yaxis.size**
This allows you to specify a size for the Y axis title.
*Default: null*

**chart.title.yaxis.font**
This allows to specify a font for the Y axis title.
*Default: null*

**chart.title.yaxis.bold**
This controls whether the Y axis title is bold or not.
*Default: true*

**chart.title.xaxis.pos**
This is multiplied with the gutter to give the position of the X axis title.
*Default: 0.25*

**chart.title.yaxis.pos**
This is multiplied with the gutter to give the position of the Y axis title.
*Default: 0.25*

## *Shadow*

**chart.shadow**
Whether a drop shadow is applied.
*Default: false*

**chart.shadow.color**
The color of the shadow.
*Default: #666*

**chart.shadow.offsetx**
The X offset of the shadow.
*Default: 3*

**chart.shadow.offsety**
The Y offset of the shadow.
*Default: 3*

**chart.shadow.blur**
The severity of the shadow blurring effect.

*Default: 3*

## *Scale*

**chart.scale.formatter**
To allow thoroughly custom formats of numbers in the scale, you can use this option to specify a function that is used by RGraph to format numbers. This function should handle ALL of the formatting. Eg:

```
function myFormatter(obj, num)
{
    return num + 'F'; // An example of formatting
}
myGraph.Set('chart.scale.formatter', myFormatter);
```

*Default: null*

**chart.scale.decimals**
The number of decimal places to display for the Y scale.
*Default: 0*

**chart.scale.point**
The character used as the decimal point.
*Default: .*

**chart.scale.thousand**
The character used as the thousand separator
*Default: ,*

**chart.units.pre**
The units that the Y axis is measured in. This string is displayed BEFORE the actual number, allowing you to specify values such as "$50".
*Default: none*

**chart.units.post**
The units that the Y axis is measured in. This string is displayed AFTER the actual number, allowing you to specify values such as "50ms".
*Default: none*

**chart.ymax**
The optional maximum Y scale value. If not specified then it will be calculated.
*Default: null (It's calculated)*

**chart.ymin**
The optional minimum Y scale value. If not specified then it will be 0.
*Default: 0*

## *Key*

**chart.key**
An array of key information.
*Default: [] (An empty array)*

**chart.key.background**
The color of the key background. Typically white, you could set this to something like rgba(255,255,255,0.7) to allow people to see things behind it.
*Default: white*

**chart.key.halign**
Instead of specifying the exact x/y coordinates, you can use this property to simply specify whether the key hould be aligned *left* or *right*.
*Default: right*

**chart.key.position**
Determines the position of the key.Either **graph** (default), or **gutter**.
*Default: graph*

**chart.key.position.x**
This allows you to specify a specific X coordinate for the key.
*Default: null*

**chart.key.position.y**
This allows you to specify a specific Y coordinate for the key.
*Default: null*

**chart.key.position.gutter.boxed**
If you have the key in gutter mode (ie horizontal), this allows you to give a background color.
*Default: true*

**chart.key.shadow**
Whether a small drop shadow is applied to the key.
*Default: false*

**chart.key.shadow.color**
The color of the shadow.
*Default: #666*

**chart.key.shadow.blur**
The extent of the blurring effect used on the shadow.
*Default: 3*

**chart.key.shadow.offsetx**
The X offset of the shadow.
*Default: 2*

**chart.key.shadow.offsety**
The Y offset of the shadow.

*Default: 2*

**chart.key.rounded**
This controls whether the corners of the key (in graph mode) are curved. If the key is gutter mode, this has no effect.
*Default: false*

**chart.key.color.shape**
This can be *square*, *circle* or *line* and controls how the color indicators in the key appear.
*Default: square*

**chart.key.linewidth**
The line width of the surrounding border on the key.
*Default: 1*

## *Interactive features*

**chart.contextmenu**
An array of context menu items.
*Default: [] (An empty array)*

**chart.tooltips**
A numerically indexed array of tooltips that are shown when a bar is clicked. These can contain HTML.
*Default: null*

**chart.tooltips.effect**
The animated effect used for showing tooltips. Can be either *fade* or *expand*.
*Default: fade*

**chart.tooltips.event**
This is the event that triggers the tooltips. It can be either *onclick* or *onmousemove*.
*Default: onclick*

**chart.tooltips.css.class**
This is the name of the CSS class the chart uses.
*Default: RGraph_tooltip*

**chart.tooltips.override**
If you wish to handle showing tooltips yourself, this should be a function object which does just that.
*Default: null*

**chart.tooltips.highlight**
When combining chartss you will need to set this option to fase.
*Default: true*

**chart.crosshairs**
If true, you will get a crosshair centering on the current mouse position.
*Default: false*

**chart.crosshairs.linewidth**
This controls the linewidth of the crosshairs.
*Default: 1*

**chart.crosshairs.color**
The color of the crosshairs.
*Default: #333*

**chart.crosshairs.hlines**
This determines whether the horizontal crosshair is shown.
*Default: true*

**chart.crosshairs.vlines**
This determines whether the vertical crosshair is shown.
*Default: true*

**chart.annotatable**
Whether annotations are enabled for the chart (ie you can draw on the chart interactively.
*Default: false*

**chart.annotate.color**
If you do not allow the use of the palette, then this will be the only colour allowed for annotations.
*Default: black*

**chart.resizable**
Defaulting to false, this determines whether your chart will be resizable. Because of the numerous event handlers this has to install code on, This feature is unlikely to work with other dynamic features (the context menu is fine however).
*Default: false*

**chart.resize.handle.background**
With this you can specify the background color for the resize handle. If you're adjusting the position of the handle then you may need this to make the handle stand out more.
*Default: null*

**chart.adjustable**
Defaulting to false, this determines whether your bar chart will be adjustable.
*Default: false*

## *Zoom*

**chart.zoom.factor**
This is the factor that the chart will be zoomed by (bigger values means more zoom)
*Default: 1.5*

**chart.zoom.fade.in**
Whether the zoomed canvas fades in or not. This also can be used to control the fade in

for the zoom in thumbnail mode.
*Default: true*

**chart.zoom.fade.out**
Whether the zoomed canvas fades out or not. This also can be used to control the fade in for the zoom in thumbnail mode.
*Default: true*

**chart.zoom.hdir**
The horizontal direction of the zoom. Possible values are: *left*, *center*, *right*
*Default: right*

**chart.zoom.vdir**
The vertical direction of the zoom. Possible values are: *up*, *center*, *down*
*Default: down*

**chart.zoom.delay**
The delay (in milliseconds) between frames.
*Default: 50*

**chart.zoom.frames**
The number of frames in the zoom animation.
*Default: 10*

**chart.zoom.shadow**
Whether or not the zoomed canvas has a shadow or not.
*Default: true*

**chart.zoom.background**
Defaulting to true, this determines whether the zoom has a dark, semi-opaque background that covers the entire web page.
*Default: true*

## *Events*

**chart.events.click**
If you want to add your own *onclick* function you can do so by assigning it to this property.
*Default: null*

**chart.events.mousemove**
If you want to add your own *onmousemove* function you can do so by assigning it to this property.
*Default: null*

## *Miscellaneous*

**chart.highlight.stroke**
If you use tooltips, this controls the colour of the highlight stroke.
*Default: black*

**chart.highlight.fill**
If you use tooltips, this controls the colour of the highlight fill.
*Default: rgba(255,255,255,0.5)*

**chart.line**
Formerly a boolean, this now stipulates a line chart object which is to be drawn on top of the bar chart.
*Default: null*

**chart.variant**
This simply stipulates whether you want a regular bar chart, a dot chart, a pyramid chart a 3D chart, a sketch or a glassy chart. Possible values are:

- bar
- dot
- pyramid
- arrow
- 3d
- sketch
- glass

The pyramid, dot, arrow, sketch and glass variants are only effective for regular bar charts - not grouped or stacked charts.
*Default: bar*

**chart.xaxispos**
The position of the X axis. This can be *bottom*, *center* or *top*.
*Default: bottom*

**chart.axis.color**
The color of the axes.
*Default: black*

**chart.grouping**
How the bars are grouped, and it should be one of: **grouped** or **stacked**
*Default: grouped*

## *Methods*

**obj.getShape(event)**

This method makes it easier to get hold of which bar has been clicked on, or hovered over. It returns an array of:

- The chart object
- The X coordinate
- The Y coordinate

- The width of the bar
- The height of the bar
- The numerical index of the bar. This corresponds (for example) to the tooltips array, and the coordinates array

The shape also includes textual indexes like this: *shape['object']* And they are:

- object
- x
- y
- width
- height
- tooltip
- index

An example usage is:

```
<canvas id="cvs" width="600" height="250">[No canvas support]</canvas>

<script src="RGraph.common.core.js"></script>
<script src="RGraph.bar.js"></script>

<script>
    myGraph = new RGraph.Bar('myCanvas', [1.2, 1.3, 1.4, 1.5,6,1.9,2,2.1,2.5]);
    myGraph.Set('chart.labels', ['John', 'Barry', 'Rich', 'Craig', 'Tom', 'Frank', 'Helen', 'Joyce', 'Fred'])
    myGraph.Draw();

    RGraph.Register(myGraph);

    myGraph.canvas.onclick = function (e)
    {       RGraph.Redraw();

        var canvas  = e.target;
        var context = canvas.getContext('2d');
        var obj     = canvas.__object__;
        var coords  = obj.getShape(e);

        if (coords) {
            var top    = coords[1];
            var left   = coords[2];
            var width  = coords[3];
            var height = coords[4];

            context.beginPath();
                context.strokeStyle = 'black';
                context.fillStyle = 'rgba(255,255,255,0.5)';
                context.strokeRect(top, left, width, height);
                context.fillRect(top, left, width, height);
            context.stroke();
            context.fill();
        }
    }
</script>
```

**obj.getShapeByX(event)**

This method is similar to the above, but only uses the X coordinate to get the relevant bar.

**obj.getValue(mixed)**

This method can be used to get the value at a particular point or at the mouse coordinates, based on the scale that is in use. Not simply the coordinates of the mouse. The argument can either be an event object (for use in event listener functions) OR a two element array consisting of the X and Y coordinates (ie when you're not necessarily in an event listener). It returns null if the mouse or coordinates are in the gutter areas. An example:

```
myChart.canvas.onclick = function (e)
{    var obj   = e.target.__object__;
   var value = obj.getValue(e);

   // ...
}
```

### *Note about the data_arr array*

Sometimes you may wish to view your data as one big array, instead of one array per dataset. In this case the *obj.data_arr* is available. This is one long array containing all of the individual data points.

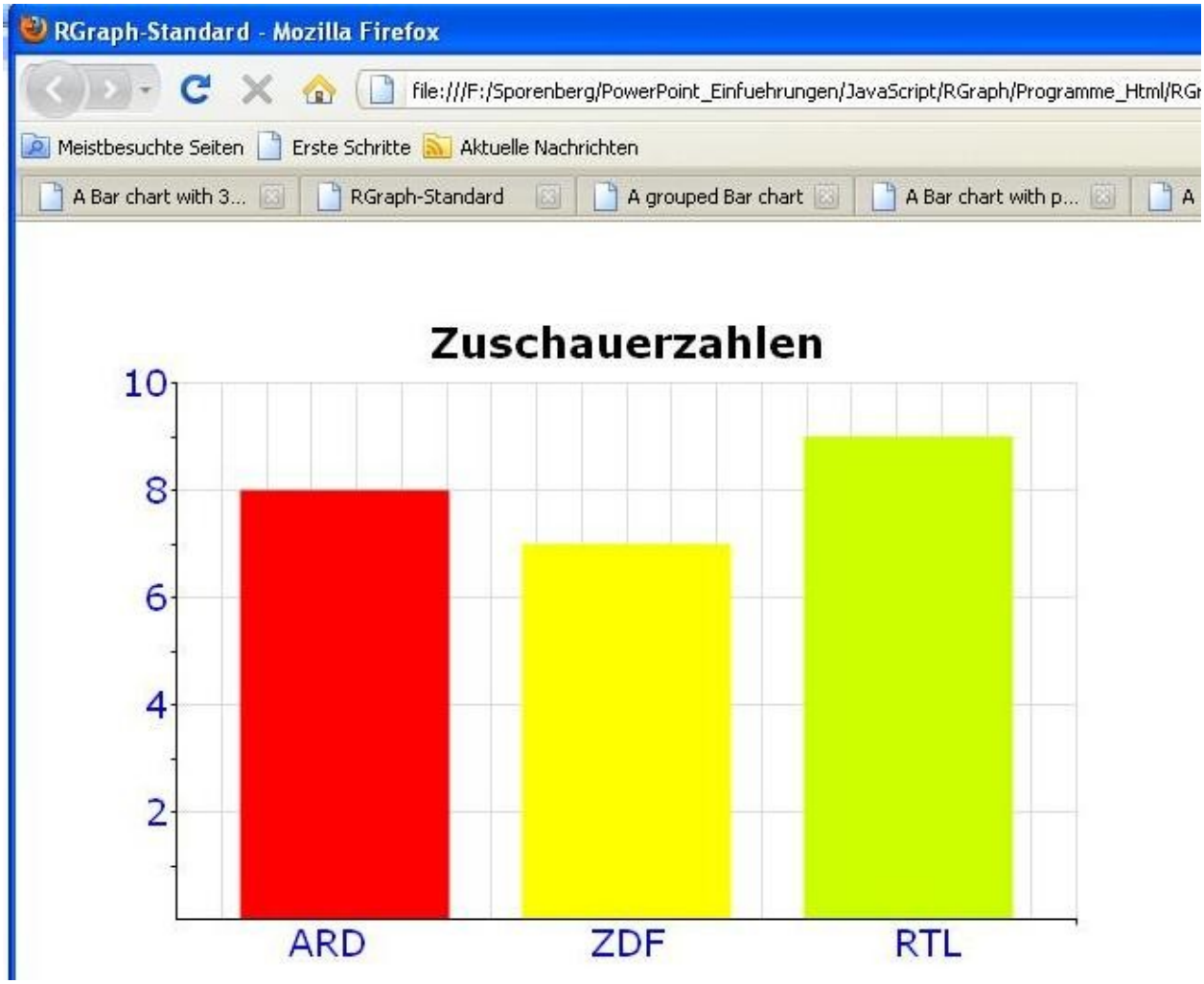
## Beispiel:

```
<html>
<head>    <title>RGRaph - Säulendiagramm</title>

<script src="RGraph.common.core.js"></script>
<script src="RGraph.bar.js"></script>

<script>
   window.onload = function ()
   {  var data = [[8,7,9]];
      var bar = new RGraph.Bar('canvas', data);
      bar.Set('chart.labels', ['ARD', 'ZDF','RTL']);
      bar.Set('chart.gutter.left', 80);
      bar.Set('chart.gutter.top', 80);
      bar.Set('chart.text.font', 'Verdana');
      bar.Set('chart.text.size', 15);
      bar.Set('chart.text.color', '#0000ff');
      bar.Set('chart.title','Zuschauerzahlen');
      bar.Set('chart.hmargin', 15);
      bar.Set('chart.hmargin.grouped', 20);
      bar.Set('chart.colors', ['#ff0000','#ffff00','#cdff00']);
      bar.Draw();    }
</script>
</head>

<body >
   <canvas id="canvas" height="400" width="600">    </canvas>
</body>
</html>
```


RGraph-Standard - Mozilla Firefox
file:///F:/Sporenberg/PowerPoint_Einfuehrungen/JavaScript/RGraph/Programme_Html/RGr
Meistbesuchte Seiten
Erste Schritte
Aktuelle Nachrichten
A Bar chart with 3...
RGraph-Standard
A grouped Bar chart
A Bar chart with p...
Zuschauerzahlen
10
8
6
4
2
ARD
ZDF
RTL