Linien-Diagramme Dokumentation – Sporenberg/08.01.2013

Note: With the Line chart, there isn't an option to have a scale on the X axis. For this you'll need a Scatter chart with *chart.line* set to true. You either use labels across the X axis or an X scale.

Example

```
<script>
    window.onload = function ()
    {
        // The data for the Line chart. Multiple lines are specified as seperate arrays.
        var data = [10,4,17,50,25,19,20,25,30,29,30,29];

        // Create the Line chart object. The arguments are the canvas ID and the data array.
        var line = new RGraph.Line("myLine", data);

        line.Set('chart.background.barcolor1', 'white');
        line.Set('chart.background.barcolor2', 'white');
        line.Set('chart.background.grid.color', 'rgba(238,238,238,1)');
        line.Set('chart.colors', ['red']);
        line.Set('chart.linewidth', 2);
        line.Set('chart.filled', true);
        line.Set('chart.hmargin', 5);
        line.Set('chart.labels', ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']);
        line.Set('chart.gutter.left', 40);

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

Properties

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

myLine.Set('name', 'value');

Background

chart.background.barcolor1
The color of the background bars (1 of 2).
Default: rgba(0,0,0,0)

chart.background.barcolor2
The color of the background bars (2 of 2).
Default: rgba(0,0,0,0)

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

chart.background.grid.color
The color of the background grid.
Default: #eee

chart.background.hbars
An array of information stipulating horizontal colored bars. You can use these to indicate limits. Eg: *myLine.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.grid.hsize
The horizontal size of the grid.
Default: 25

chart.background.grid.vsize
The vertical size of the grid.
Default: 25

chart.background.grid.width
The width of the background grid.
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.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

chart.backdrop
When enabled this specifies that the line(s) will have a backdrop effect. You can control the transparency with the other backdrop settings).
Default: false

chart.backdrop.size
This controls the size/extent of the backdrop effect.
Default: 30

chart.backdrop.alpha
This controls how much transparency the backdrop effect has. It can go from 0 - 1.
Default: 0.2

Labels and text

chart.labels.above
Whether the values are shown in labels drawn above the line.
Default: false

chart.labels.above.size
The size of the labels which are drawn above the line.
Default: 8

chart.labels
An array of the X labels for the chart.
Default: [] (An empty array)

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.invert
Reverses the Y axis so that 0 is at the top, instead of the bottom.
Default: false

chart.ylabels.count
A value (*1, 3, 5 or 10*) that controls how many Y labels there are.
Default: 5

chart.ylabels.inside
This controls whether the Y labels are drawn inside the Y axis or not. If your labels are large, this may help.
Default: false

chart.ylabels.inside.color
If the Y labels are to be drawn inside the Y axis, this is used as the background color.
Default: rgba(255,255,255,0.5)

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

chart.xlabels.inside
This controls whether the X labels are drawn inside the X axis or not. By using this you can significantly reduce the size of the gutter needed.
Default: false

chart.xlabels.inside.color
If the X labels are to be drawn inside the X axis, this is used as the background color.
Default: rgba(255,255,255,0.5)

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

chart.text.angle
The angle of the horizontal text labels (at the bottom of the chart). Previously this could be 0, 45 or 90, but now (31st July 2010) this can be any angle.
Default: 0 (Horizontal)

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

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

Margins

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

chart.hmargin
The size of the horizontal margin. This is on the inside of the axes.
Default: 0

Colors

chart.colors
An array of line colors.
Default: ['#f00', '#0f0', '', '#00f', '#f0f', '#ff0', '#0ff'] 9

chart.colors.alternate

Set this to true if you want your line color(s) to be alternated. See note
Default: false

chart.fillstyle
A single color or an array of colors that filled line charts will use.

Important: This used to be a string, and still can be, but can now also be an array.
Default: null

chart.filled
Whether the area under the line is filled or not. This looks best when there is no horizontal margin.

Note: When showing multiple lines the values are additive by default. This means that if you have two lines they will be "stacked" on top of each other. If this is not the desired behaviour then you can set the option below to false.
Default: false

chart.filled.accumulative
When showing multiple filled lines the values are by default accumulative (ie added to each other). If this is not the desired behaviour, then you can set this property to *false* to have them drawn "non-accumulatively".

Note: If you use fully opaque colors in conjunction with this option set to false it's feasible that you might not see one or more of the lines, or parts of the line. If you want to see all of the lines you should leave this option set to its default *true* setting.
Default: true

chart.filled.range
This is useful for indicating a range. Exactly two datasets are required, with the space between them filled. This is useful for indicating a range.
Default: false

Shadow

chart.shadow
If true a shadow will be applied to the line.
Default: false

chart.shadow.color
The color of the shadow. As well as a single color definition, this can also be an array of colors. This means that if you have multiple lines on your chart, each can have a different shadow color.
Default: rgba(0,0,0,0.5)

chart.shadow.offsetx
The X offset in pixels for the shadow.
Default: 3

chart.shadow.offsety
The Y offset in pixels for the shadow.
Default: 3

chart.shadow.blur
The severity of the shadow blurring effect.
Default: 3

Interactive features

chart.tooltips
These are tooltips for the line(s). It should be an array of tooltips. If you have multiple lines, simply pass multiple arrays to the Set() method.
Default: [] (An empty array)

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

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
If you don't want/need the chart to be highlighted and thus avoid redrawing, then set this to false.
Default: true

chart.tooltips.hotspot.xonly
Set this to true if you want the tooltips to be triggered by the mouse X position only.
Default: false

chart.tooltips.coords.page
If you want the tooltips positioned at the event.pageX and event.pageY coordinates then set this to true.
Default: false

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.contextmenu
An array of context menu items. Unlike the bar chart, you CAN have context menus at the same time as tooltips.
Default: [] (An empty array)

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 color 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 chart will be adjustable (click a point and drag it).
Default: false

Titles

chart.title
The title of the chart.
Default: none

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

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

chart.title.yaxis.align
Instead of using the option above you can instead use this option, specifying *left* or *right*.
Default: left

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*. By default the key is positioned on the opposite side to the Y axis.
Default: null

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

chart.key.interactive
The Line chart supports interactive keys - so you can click on a key element and RGraph will highlight the appropriate line, whilst fading the rest of the chart.
Default: false

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.units.post
The units (if any) that the Y axis is measured in (gets appended to the number)
Default: none

chart.units.pre
The units (if any) that the Y axis is measured in (gets preppended to the number)
Default: none

chart.scale.decimals
Determines the precision of the numbers used as the 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.scale.round
Whether to round the maximum scale value up or not. This will produce slightly better scales in some instances.
Default: null

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

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

chart.outofbounds
Normally, out-of-bounds values are not drawn. By setting this to *true* you can change this behaviour.
Default: false

Axis properties

chart.numxticks
The number of X tickmarks.
Default: null (linked to number of datapoints)

chart.numyticks
The number of Y tickmarks.
Default: 10

chart.ticksize
The size of the tick marks. This only affects certain styles of tickmarks.
Default: 3

chart.tickdirection
Whether the ticks are above or below the axis.
Default: -1 (-1 is below, 1 is above)

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

chart.xaxispos
The position of the X axis. It can be either *bottom* or *center*.
Default: bottom

chart.yaxispos
Specifies the Y axis position. Can be *left* or *right*.
Default: left

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

chart.axesontop
A minor option, this sets the axes to be redrawn after the chart has been drawn. This is only useful in a certain set of circumstances - the chart is filled and the line width is small.
Default: false

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

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.tickmarks
What kind of tickmarks to use on the chart. This can be:

- dot
- circle
- filledcircle
- endcircle
- square
- endsquare
- filledsquare
- filledendsquare
- tick
- halftick
- endtick
- cross
- borderedcircle (same as dot)
- arrow
- filledarrow

Note that "arrow" and "filledarrow" look better with a thinner (1 or 2) linewidth setting. Also note that now (10th August 2010) as well as a string, this can be an array of different tickmark styles.
Default: null

chart.tickmarks.dot.color
This is the color of the BORDER around the dot/borderedcircle style tickmarks.
Default: #fff

chart.tickmarks.linewidth
This is the width of the line used when drawing the tickmarks. By default this is the same as the chart.linewidth setting.
Default: null

chart.stepped
Draws the line as stepped. Useful for showing stock performance for example.
Default: false

chart.linewidth
The width of the line (ie the actual line on the chart). Note: If your line is stepped and filled, and you don't want a trailing line indicating the last value, you can set this to zero.
Default: 1

chart.variant
At present this can only be *3d*, and gives a small 3D effect.

Default: null

chart.animation.unfold.x
This is used by the Unfold Line chart animation and dictates whether the X value is unfolded.
Default: false

chart.animation.unfold.y
This is used by the Unfold Line chart animation and dictates whether the Y value is unfolded.
Default: true

chart.animation.unfold.initial
This property can be used to set the initial factor for the Unfold animation. Setting this to a value less than one will cause the line to expand outwards, whilst a value greater than one will cause the line to contract towards the correct values.
Default: 2

chart.chromefix
Since version 6, Chrome has had a shadow bug, which becomes apparent when you use shadow blurring. This value defaults to true and means that RGraph will skirt the bug with a small fix.
Default: true

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.curvy
If you prefer - this draws a more cuvy chart. Because of the curves these charts are not exact, and the inaccuracy increases with a greater curve factor (controlled by the next property). Being new, this property may not work with all the line chart features and should be used with caution.
Default: false

chart.curvy.factor
This controls the severity of the curves when the above property is used. It can be a decimal from 0 (not curvy at all) to 0.5 (quite curvy).
Default: 0.25

Methods

obj.getShape(event)

This method makes it easier to get hold of which point on the Line chart has been hovered over. It returns an array of:

- The chart object
- The X coordinate
- The Y coordinate
- The numerical index of the point. 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
- tooltip
- index
- index_adjusted
- dataset

An example usage is:

```
<canvas id="cvs" width="600" height="300">[No canvas support]</canvas>

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

<script>
    myGraph = new RGraph.Line('cvs', [10,4,2,4,1]);
    myGraph.Set('chart.hmargin', 10);
    myGraph.Set('chart.tickmarks', 'endcircle');
    myGraph.Set('chart.labels', ['Fred','John','Kev','Lou','Pete']);
    myGraph.Draw();


    RGraph.Register(myGraph);

    myGraph.canvas.onmousemove = function (e)
    {
        RGraph.FixEventObject(e);

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

        // This is the method which simplifies getting coordinates
        var point = obj.getShape(e);


        if (point) {

            canvas.style.cursor = 'pointer';

            // Is this the same tooltip as the one (if any) that's already being shown
            if (RGraph.Registry.Get('chart.tooltip') && RGraph.Registry.Get('chart.tooltip').__index__ ==
point[3]) {
                return;
            }

            // Start afresh
```

```
            RGraph.Redraw();

            // Show the tooltip
            RGraph.Tooltip(canvas, obj.Get('chart.labels')[point[3]], e.pageX, e.pageY, point[3]);

            // Highlight the point
            context.strokeStyle = 'gray';
            context.fillStyle = 'white';
            context.beginPath();
            context.moveTo(point[1], point[2]);
            context.arc(point[1], point[2], 2, 0, 6.26, 0);
            context.stroke();
            context.fill();

            return;
        }

        canvas.style.cursor = 'default';
    }

    window.onclick = function ()
    {
        RGraph.Redraw();
    }
</script>
```

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);

    // ...
}
```

Combining the Line and Bar charts

You can combine the Bar and Line charts with. In the same vein, you can have Y axes on both the left and right sides.

Alternative colors

Instead of a string stipulating the color, each element of the colors array can itself be a two element array, stipulating the up color, and the down color. To use alternating colors you must also stipulate the alternate property:

```
myLine.Set('chart.colors.alternate', true);
myLine.Set('chart.colors', ['red', ['blue', 'yellow'], 'green]);
```

Accumulative filled Line charts

The default behaviour of filled Line charts is to "stack" the lines on top of each other. This allows them all to be totally visible, no matter what (unless a line has a zero value of course). If this is not desired, then there is an option (*chart.filled.accumulative* - true/false) to change this behaviour so the lines are plotted "as-is". Keep in mind that if you set this option to false (ie the Lines are plotted as-is) it may be wiser to use semi-transparent colors or some parts of data sets (or even entire data sets) may be hidden by others.

Custom tickmarks

If none of the available tickmark styles are suitable, you can instead specify a function object that draws the tickmark, enabling you to draw the tickmark yourself. For example:

```
<script>
    line.Set('chart.tickmarks', myTick);

    /**
    * The function that is called once per tickmark, to draw it
    *
    * @param object obj   The chart object
    * @param array  data  The entire line data
    * @param number value The individual points value
    * @param number index The current index, in the data array
    * @param number x     The X coordinate
    * @param number y     The Y coordinate
    * @param string color The color of the line
    * @param number prevX The previous X coordinate
    * @param number prevY The previous Y coordinate
    */
    function myTick (obj, data, value, index, x, y, color, prevX, prevY)
    {
        // Draw your custom tick here
    }
</script>
```

The coords2 array

An alternative method of indexing the chart coordinates is available in *obj.coords2*. With this array, all of the first lines coordinates are available in *obj.coords2[0]*, the second lines coordinates in *obj.coords2[1]* and so on.

The __index2__ property on tooltips

When showing tooltips, one property of the tooltip is *.__index2__*. This is the index that pertains to the individual dataset. In a function called from the *ontooltip* event you can access it like this:

```
function myFunc (obj)
{
    var idx = RGraph.Registry.Get('chart.tooltip').__index2__;
}
RGraph.AddCustomEventListener(myObject, 'ontooltip', myFunc  );
```

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.

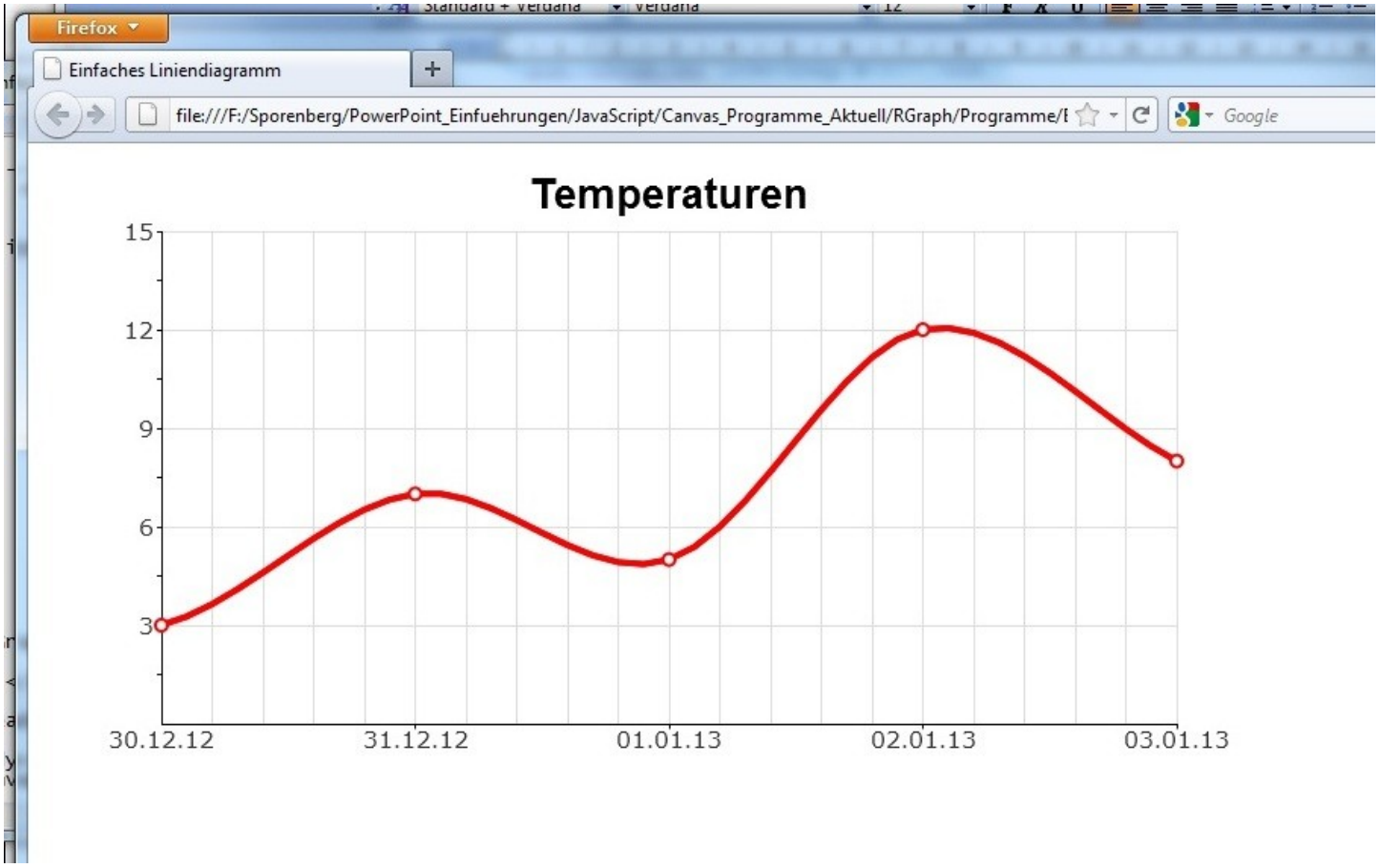
Beispiel:

```
<html><head>
<script src="../libraries/RGraph.common.core.js" ></script>
<script src="../libraries/RGraph.common.dynamic.js" ></script>
<script src="../libraries/RGraph.line.js" ></script>
<!--[if lt IE 9]><script src="../excanvas/excanvas.js"></script><![endif]-->

<title>Einfaches Liniendiagramm</title>

<script>
  window.onload = function ()
      { var line = new RGraph.Line('canvas', [3,7,5,12,8]);
        line.Set('chart.labels', ['30.12.12','31.12.12','01.01.13','02.01.13','03.01.13']);
        line.Set('chart.title.font','Arial'); // Die Schriftart für den Titel wird vereinbart
        line.Set('chart.title.size',20); // Die Schriftgröße für den Titel wird vereinbart
        line.Set('chart.title','Temperaturen');
        line.Set('chart.curvy', true);  //  Die Verbindung der Punkte wird geglättet
        line.Set('chart.tickmarks', true);
        line.Set('chart.linewidth', 4);
        line.Set('chart.gutter.left', 80);
        line.Set('chart.gutter.top', 50);
        line.Set('chart.text.font', 'Verdana');
       // Die Schriftart für den Text wird vereinbart
        line.Set('chart.text.size', 11); // Die Schriftgröße für den Text wird vereinbart
        line.Set('chart.text.color', '#343434');
        line.Set('chart.tickmarks.dot.color','#0000ff');
        line.Set('chart.tickmarks', 'circle');
        line.Set('chart.tickmarks.size', 40);
        line.Draw()         }
</script>
</head>

<body>
<canvas id="canvas" width="800" height="400">Keine Canvas-
Unterstützung</canvas>
</body>
</html>
```


Firefox
Einfaches Liniendiagramm
file:///F:/Sporenberg/PowerPoint_Einfuehrungen/JavaScript/Canvas_Programme_Aktuell/RGraph/Programme/I
Google
Temperaturen
15
12
9
6
3
30.12.12
31.12.12
01.01.13
02.01.13
03.01.13