ChartsEngine 1.1.1

How to use

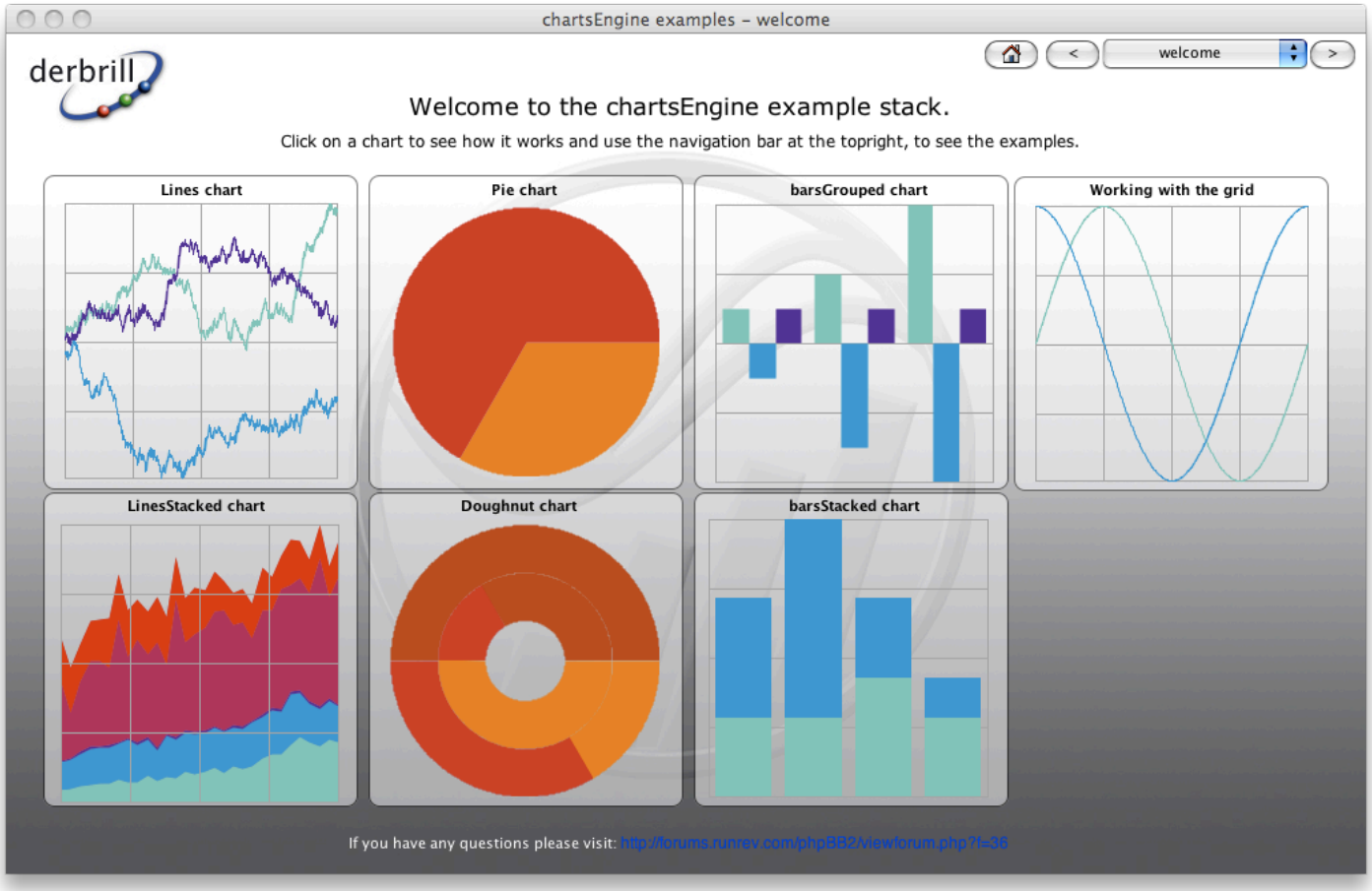
ChartsEngine is a powerful, yet lightweight library to create charts in a revolution stack. It has been optimized to render data on the fly in high speed. It will extend Revolution's messagepath and makes it's scripts available in any location in your Revolution application.

To use it, make sure it is loaded into memory. Open the messagebox and type:

start using stack "chartsengine"

In the IDE you also can double click the chartsEngine stack to start using it. The following chartTypes are available:

- Lines
- LinesStacked
- Bars
- BarsStacked
- Pie
- Doughnut



Basic explanation

ChartsEngine relies on setting properties for your charts. Most of these properties are "canset" properties. That means, if you do not explicitly set them, chartsEngine defaults to standard values. Data is organized in CR delimited lists of items. The default itemDelimiter is comma. The default chart type is lines. This type lets you handle large amounts of data, while rendering very fast. Which chart type is best used in your projects depends on the data you want to display. While the line and bar types are not limited in amount of data they can process, the pie and doughnut charts are limited to 300 entries per "ring".

Creating a chart

chartsEngine creates charts by Revolution script. The following line of Code:

chartsCreateChart "nameOfChart"

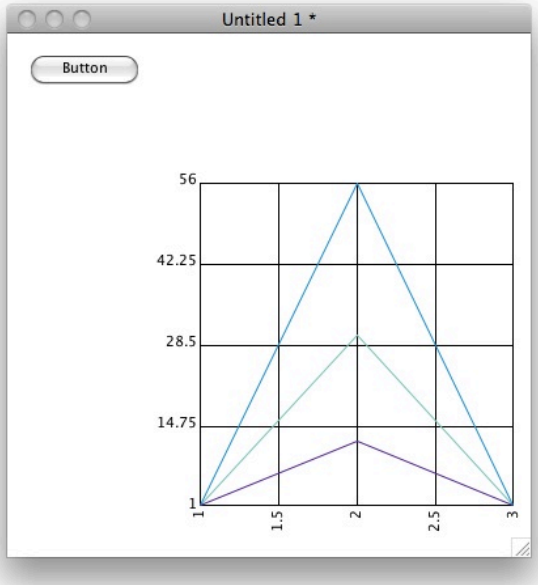
Will create a group with the name "nameOfChart" for you. The result contains a reference to the group that you can store in a variable for later reuse. Now you can set a few custom properties for this group to render the chart inside of that group. These properties are organized in the custompropertyset "charts" of the group that has been created. The most common custom property you will set is the

charts["data"] of the group.

To finally render the chart you send the command chartsRefreshChart to the group, or use chartsRefresh as a command with an additional parameter containing a reference to the group.

A complete script to create a chart can look like this:

```
on mouseUp
  local tchart
  chartsCreateChart
  put the result into tchart
  set the charts["data"] of tchart to "1,1,1" & cr & "30,56,12" & cr & "1,1,1"
  chartsRefresh tChart
end mouseUp
```



The chartsCreateChart command can take additional Parameters. You can specify the name of the group being created and the rectangle the chart should be drawn in. Possible uses are:

chartsCreateChart "myChart"
chartsCreateChart "myChart", the rectangle of this card
chartsCreateChart "myChart", tLeft,tTop,tRight,tBottom

Switching the style

To switch the style of a chart you simply set a property and refresh the chart afterwards. Choose one of these lines to switch the style:

set the charts["chartStyle"] of group "myChart" to "lines"
set the charts["chartStyle"] of group "myChart" to "linesStacked"
set the charts["chartStyle"] of group "myChart" to "barsGrouped"
set the charts["chartStyle"] of group "myChart" to "barsStacked"
set the charts["chartStyle"] of group "myChart" to "pie"
set the charts["chartStyle"] of group "myChart" to "doughnut"

and use chartsRefresh the long ID of grp "myChart" to refresh the chart afterwards.

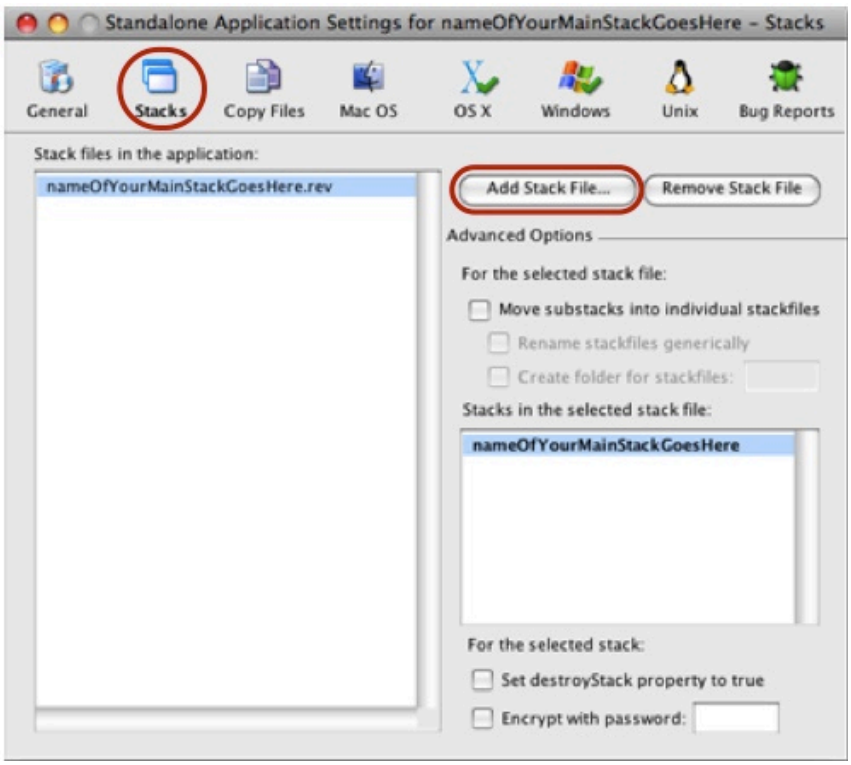
Preparing for distribution

chartsEngine is a library. To make it work in a standalone application, it must be loaded and put in use. Make sure your standalone is able to find the

library. You can do this by either embedding chartsEngine as a substack, or let the standalonebuilder handle that. To do the former: Open the messagebox and type:

set the mainstack of stack "chartsEngine" to "nameOfYourMainStack"
save stack "YourMainStackNameGoesHere"

If you want to use the standalone application setting, klick on "stacks" first, then on "Add stack File..." and locate chartsEngine on your Harddisk.



Setting chart properties

While you can use just a few lines of script to create your charts, the library is very flexible in regards of settings you can make. A chart group owns the custom property set charts, which stores all relevant settings for your chart. You can set these properties by script using array notation.

set the charts[tProperty] of group "myChart" to tValue

Attention! - Make sure to quote the property name in your scripts e.g.

set the charts["data"] of group "myChart" to 100,200&cr&30,50

How to use

Comments:
ChartsEngine is a powerful, yet lightweight library to create charts in a revolution stack. It has been optimized to render data on the fly in high speed. It will extend Revolution's messagepath and makes it's scripts available in any location in your Revolution application.

To use it, make sure it is loaded into memory. Open the messagebox and type:

start using stack "chartsengine"

In the IDE you also can double click the chartsEngine stack to start using it. The following chartTypes are available:

- Lines
- LinesStacked
- Bars
- BarsStacked
- Pie
- Doughnut

Basic explanation

ChartsEngine relies on setting properties for your charts. Most of these properties are "canset" properties. That means, if you do not explicitly set them, chartsEngine defaults to standard values. Data is organized in CR delimited lists of items. The default itemDelimiter is comma. The default chart type is lines. This type lets you handle large amounts of data, while rendering very fast. Which chart type is best used in your projects depends on the data you want to display. While the line and bar types are not limited in amount of data they can process, the pie and doughnut charts are limited to 300 entries per "ring".

Creating a chart

chartsEngine creates charts by Revolution script. The following line of Code:

chartsCreateChart "nameOfChart"

Will create a group with the name "nameOfChart" for you. The result contains a reference to the group that you can store in a variable for later reuse. Now you can set a few custom properties for this group to render the chart inside of that group. These properties are organized in the custompropertyset "charts" of the group that has been created. The most common custom property you will set is the

charts["data"] of the group.

To finally render the chart you send the command chartsRefreshChart to the group, or use chartsRefresh as a command with an additional parameter containing a reference to the group.

A complete script to create a chart can look like this:

```
on mouseUp
  local tchart
  chartsCreateChart
  put the result into tchart
  set the charts["data"] of tchart to "1,1,1" & cr & "30,56,12" & cr & "1,1,1"
  chartsRefresh tChart
end mouseUp
```

The chartsCreateChart command can take additional Parameters. You can specify the name of the group being created and the rectangle the chart should be drawn in. Possible uses are:

```
chartsCreateChart "myChart"
chartsCreateChart "myChart", the rectangle of this card
chartsCreateChart "myChart", tLeft,tTop,tRight,tBottom
```

Switching the style

To switch the style of a chart you simply set a property and refresh the chart afterwards. Choose one of these lines to switch the style:

```
set the charts["chartStyle"] of group "myChart" to "lines"
set the charts["chartStyle"] of group "myChart" to "linesStacked"
set the charts["chartStyle"] of group "myChart" to "barsGrouped"
set the charts["chartStyle"] of group "myChart" to "barsStacked"
set the charts["chartStyle"] of group "myChart" to "pie"
set the charts["chartStyle"] of group "myChart" to "doughnut"
```

and use chartsRefresh the long ID of grp "myChart" to refresh the chart afterwards.

Preparing for distribution

chartsEngine is a library. To make it work in a standalone application, it must be loaded and put in use. Make sure your standalone is able to find the library. You can do this by either embedding chartsEngine as a substack, or let the standalonebuilder handle that. To do the former: Open the messagebox and type:

```
set the mainstack of stack "chartsEngine" to
"nameOfYourMainStackGoesHere"
save stack "nameOfYourMainStackGoesHere"
```

If you want to use the standalone application setting, klick on "stacks" first, then on "Add stack File..." and locate chartsEngine on your Harddisk.

Setting chart properties

While you can use just a few lines of script to create your charts, the library is very flexible in regards of settings you can make. A chart group owns the custom property set charts, which stores all relevant settings for your chart. You can set these properties by script using array notation.

```
set the charts[tProperty] of group "myChart" to tValue
```

Attention! - Make sure to quote the property name in your scripts e.g.

```
set the charts["data"] of group "myChart" to 100,200&cr&30,50
```

General Info

The bars stacked type should not be used with data that's all positive (and zero), or all negative (and zero) per line. Weird results would come out of a data line like "3,-2,-2". But of course you can mix negative and positive lines freely.

The pie and dougnut types are limited to 300 datasets, and the doughnut type maximum ring number is 50.

When using the Values with points by setting the charts["showPoints"] to true, their size varies widely based on which style they're set to. So for example, setting charts["pointsStyle"] to "triangle" creates a *visually* much smaller triangle then the default "Circle".

Properties

Summary:
There are many properties that govern how a chart might look. Here is a list of all of them:

Comments:
charts["data"]
charts["dataIncludesX"]
charts["chartItemDel"]
charts["chartStyle"]
charts["antialiased"]
charts["chartBackgroundColor"]
charts["chartColors"]
charts["lineSizes"]
charts["showLegend"]
charts["legendNames"]
charts["legendTextFont"]
charts["legendTextColor"]
charts["legendTextSize"]
charts["gridX"]
charts["gridY"]
charts["gridInFront"]
charts["showGridXDescription"]
charts["showGridYDescription"]

charts["minX"] charts["maxX"] charts["minY"] charts["maxY"]
charts["gridXDescriptionOrientation"]
charts["gridXNumberFormat"] charts["gridYNumberFormat"]
charts["xConvertsTo"]
charts["xPrefix"]
charts["xPostfix"]
charts["descriptionTextFont"]
charts["descriptionTextSize"]
charts["descriptionTextColor"]
charts["gridColor"]
charts["gridXFillColor"] charts["gridYFillColor"]
charts["gridXFillAltColor"] charts["gridYFillAltColor"]
charts["gridXInk"] charts["gridYInk"]
charts["caption"]
charts["captionTextFont"]
charts["captionTextSize"]
charts["captionTextColor"]
charts["captionAlign"]
charts["showValues"]
charts["valuesNumberFormat"]
charts["valuesTextFont"]
charts["valuesTextSize"]
charts["valuesTextColor"]
charts["valuesAlign"]
charts["valuesPrefix"] charts["valuesPostfix"]
charts["showPoints"]
charts["pointsColor"]
charts["pointsSize"]
charts["pointsStyle"]

charts["data"]

Summary:
Container for chartData. Should be a cr delimited list of items.

Examples:
set the charts["data"] of grp "myChart" to "1,2,3"&cr&"3,2,4"
set the charts["data"] of grp "myChart" to tData

Comments:
The charts["data"] is a container that holds your raw data. This data is organised as a set of lines of numeric items. The first item can be used to describe the position of the dataset on the x-axis, thus allowing the x-axis to scale correctly for missing datapoints. to do that, use charts ["dataIncludesX"].

charts["dataIncludesX"]

Summary:
Governs if the first item of each line is used to calculate the x-Axis.

Examples:
set the charts["dataIncludesX"] of grp "myChart" to true
set the charts["dataIncludesX"] of grp "myChart" to false

Comments:
In order to scale the x-axis correctly if dataIncludesX is used, chartsEngine expects a numeric value as the first item of each line. However, the labeling can be converted to everything the Livecode 'convert' command can understand by using charts["xConvertsTo"].

charts["chartItemDel"]

Summary:
The item delimiter for charts["data"].
By default, or if set to empty the default is comma: ,

Examples:
set the charts["chartItemDel"] of grp "myChart" to TAB
set the charts["chartItemDel"] of grp "myChart" to ";"

charts["chartStyle"]

Summary:
Determines the chartType that is being drawn. Possible entries:
lines, linesStacked, barsGrouped, barsStacked, pie, doughnut, <empty>

Examples:
set the charts["chartStyle"] of grp "myChart" to "lines"
set the charts["chartStyle"] of grp "myChart" to "linesStacked"
set the charts["chartStyle"] of grp "myChart" to "barsGrouped"
set the charts["chartStyle"] of grp "myChart" to "barsStacked"
set the charts["chartStyle"] of grp "myChart" to "pie"
set the charts["chartStyle"] of grp "myChart" to "doughnut"

Comments:
The quickest chart types to draw are the lines and the linesStacked types. These types are not limited in the amount of data they can draw and render rather quickly. The barsGrouped and barsStacked types aren't limited in the amount of data they can display, however depending on the other properties you set might yield unexpected results. The pie and dougnut types are limited to 300 datasets.

charts["antialiased"]

Summary:
This determines if the chart is drawn antialiased or not.

Examples:
set the charts["antialiased"] of grp "myChart" to true
set the charts["antialiased"] of grp "myChart" to false

Comments:
While antialiasing looks prettier than not using it, the downside is, that it adds non neglectable overhead to the rendering time of a chart. If you are rendering larger datasets for bar, pie or doughnut types you might notice a lag in drawing, that might be reduced by setting the antialiased of the chart to false. Attention: A bug in the Revolution engine prior to 3.5 prevents antialiased graphics to draw if they are too big. The library tries to fall back to non antialiased drawing if that happens. This adds quite a bit of overhead to the rendering time for older rev versions.

charts["chartBackgroundColor"]

Summary:
Accepts any color reference, or empty.

Examples:
set the charts["chartBackgroundColor"] of grp "myChart" to "red"
set the charts["chartBackgroundColor"] of grp "myChart" to empty
set the charts["chartBackgroundColor"] of grp "myChart" to 125,30,45

Comments:
If the chartBackgroundcolor of a chart is empty, its opaque is automatically set to false. As soon as you set a colorvalue, the groups opaque will be set to true and the background will be filled with said color.

charts["chartColors"]

Summary:
Accepts a list of color references, or empty.

Examples:
set the charts["chartColors"] of grp "myChart" to
"red"&cr&"127,0,40"&cr&"#eeeeff"
set the charts["chartColors"] of grp "myChart" to tList

Comments:
If the chartColors of a chart is empty, default colors will be used. If you specify less colors than lines needed to be drawn, it will loop through the list.

charts["lineSizes"]

Summary:
Accepts a list of numbers, or empty. Linesize for each chart element is set by this property.

Examples:
set the charts["lineSizes"] of grp "myChart" to "1"&cr&"2"&cr&"3"

charts["showLegend"]

Summary:
Governs if the legend of the chart is shown or not.

Examples:
set the charts["showLegend"] of grp "myChart" to true
set the charts["showLegend"] of grp "myChart" to false

charts["legendNames"]

Summary:
Accepts a list of text strings. They will be displayed as the legend.

Examples:
set the charts["legendNames"] of grp "myChart" to
"USA"&cr&"Europe"&cr&"Asia"

Comments:
If left empty, the string 'dummy legend' will be used if you choose to display the legend.

charts["legendTextFont"]

Summary:
Accepts any text. Make sure you use a propper font name.

Examples:
set the charts["legendTextFont"] of grp "myChart" to "Verdana"

charts["legendTextColor"]

Summary:
Accepts any color reference.

Examples:
set the charts["legendTextColor"] of grp "myChart" to "red"
set the charts["legendTextColor"] of grp "myChart" to "127,0,256"

charts["legendTextSize"]

Summary:
Accepts a number.

Examples:
set the charts["legendTextSize"] of grp "myChart" to 12

charts["gridX"]

Summary:
Accepts a number. Governs how many lines are drawn into the xGrid for line types or which bars will be labeled for bar types.

Examples:
set the charts["gridX"] of grp "myChart" to 3 -- draw 3 lines into the grid for lineTypes. Label every 3rd bar
set the charts["gridX"] of grp "myChart" to 0 -- no xGrid is drawn; barCharts is not labeled

charts["gridY"]

Summary:
Governs how many lines are drawn into the yGrid.

Examples:
set the charts["gridY"] of grp "myChart" to 3
set the charts["gridY"] of grp "myChart" to 0 -- no yGrid is drawn

charts["gridInFront"]

Summary:
Governs if the grid is drawn on top of the graphed data or not.

Examples:
set the charts["gridInFront"] of grp "myChart" to true
set the charts["gridInFront"] of grp "myChart" to false

charts["showGridXDescription"]

Summary:
If true, xAxis will be labeled at the lines drawn by gridX property.

Examples:
set the charts["showGridXDescription"] of grp "myChart" to true
set the charts["showGridXDescription"] of grp "myChart" to false

charts["showGridYDescription"]

Summary:
If true, yAxis will be labeled at the lines drawn by gridY property.

Examples:
set the charts["showGridYDescription"] of grp "myChart" to true
set the charts["showGridYDescription"] of grp "myChart" to false

charts["minX"] charts["maxX"] charts["minY"] charts["maxY"]

Summary:
If set to a number, that value will be used as a minimum value for the xAxis.
If set to a number, that value will be used as a maximum value for the xAxis.
If set to a number, that value will be used as a minimum value for the yAxis.
If set to a number, that value will be used as a maximum value for the yAxis.

Examples:
set the charts["minX"] of grp "myChart" to -2
set the charts["maxX"] of grp "myChart" to 500

set the charts["minY"] of grp "myChart" to 0
set the charts["maxY"] of grp "myChart" to 500

Comments:
Setting these properties allow you to scale the amplitude for a chart and also to scroll through a dataset. If values are outside the boundaries of the minY and maxY values, the chart will clip though.

charts["gridXDescriptionOrientation"]

Summary:
Governs if the gridX labels are rendered horizontally or vertically.

Examples:
set the charts["gridXDescriptionOrientation"] of grp "myChart" to "normal" -- horizontally
set the charts["gridXDescriptionOrientation"] of grp "myChart" to "right" -- vertically, right aligned
set the charts["gridXDescriptionOrientation"] of grp "myChart" to "left" --

vertically, left aligned

Comments:
Default value (if not set or set to empty) is "right"

charts["gridXNumberFormat"] charts ["gridYNumberFormat"]

Summary:
Numberformat for x labels
Numberformat for y labels

Examples:
set the charts["gridYNumberFormat"] of grp "myChart" to "#.000"
set the charts["gridXNumberFormat"] of grp "myChart" to "000.0"

charts["xConvertsTo"]

Summary:
Lets you convert the labeling of the x-axis to anything the convert command can handle.

Examples:
set the charts["xConvertsTo"] of grp "myChart" to "english date"
set the charts["xConvertsTo"] of grp "myChart" to "system date and long time"

Comments:
If you want to display dates or timestamps as labels for the x-axis, you can use this property. chartsEngine expects the first item to be in seconds format then. This will only work, if the charts["dataIncludesX"] of the group is true.

charts["xPrefix"]

Summary:
Lets you add a string before the x-axis label.

Examples:
set the charts["xPrefix"] of grp "myChart" to "abc"

charts["xPostfix"]

Summary:
Lets you add a string after the x-axis label.

Examples:
set the charts["xPostfix"] of grp "myChart" to "kg"

charts["descriptionTextFont"]

Summary:
Sets the font for x and y labels

Examples:
set the charts["descriptionTextFont"] of grp "myChart" to "Verdana"
set the charts["descriptionTextFont"] of grp "myChart" to line 15 of the fontnames

charts["descriptionTextSize"]

Summary:
Sets the textSize for x and y labels.

Examples:
set the charts["descriptionTextSize"] of grp "myChart" to 12

charts["descriptionTextColor"]

Summary:
Sets the textColor for x and y labels.

Examples:
set the charts["descriptionTextColor"] of grp "myChart" to "red"
set the charts["descriptionTextColor"] of grp "myChart" to "127,80,0"

charts["gridColor"]

Summary:
Sets the color of the gridLines.

Examples:

set the charts["gridColor"] of grp "myChart" to "dark gray"
set the charts["gridColor"] of grp "myChart" to "r127,233,40"
set the charts["gridColor"] of grp "myChart" to "#eeff00"

charts["gridXFillColor"] charts["gridYFillColor"]

Summary:
Choose the first color of the alternating rows of the xGrid. Will not be filled if empty.
Choose the first color of the alternating rows of the yGrid. Will not be filled if empty.

Examples:
set the charts["gridXFillColor"] of grp "myChart" to "dark gray"
set the charts["gridYFillColor"] of grp "myChart" to "127,233,40"
set the charts["gridXFillColor"] of grp "myChart" to "#eeff00"

charts["gridXFillAltColor"] charts["gridYFillAltColor"]

Summary:
Sets the color of the second alternating row of the xGrid. Will not be filled if empty.
Sets the color of the second alternating row of the yGrid. Will not be filled if empty.

Examples:
set the charts["gridXFillAltColor"] of grp "myChart" to "dark gray"
set the charts["gridYFillAltColor"] of grp "myChart" to "127,233,40"
set the charts["gridXFillAltColor"] of grp "myChart" to "#eeff00"

charts["gridXInk"] charts["gridYInk"]

Summary:
Sets the blend mode for the xGrid.
Sets the blend mode for the yGrid.

Examples:
set the charts["gridYInk"] of grp "myChart" to "blend"
set the charts["gridXInk"] of grp "myChart" to "srcCopy"
set the charts["gridXInk"] of grp "myChart" to "notSrcAnd"

charts["caption"]

Summary:
Set the headline for the chart.

Examples:
set the charts["caption"] of grp "myChart" to "My beautiful chart"

charts["captionTextFont"]

Summary:
Sets the font for the chart header.

Examples:
set the charts["captionTextFont"] of grp "myChart" to "Verdana"
set the charts["captionTextFont"] of grp "myChart" to line 15 of the fontnames

charts["captionTextSize"]

Summary:
Sets the textSize for the chart header.

Examples:
set the charts["captionTextSize"] of grp "myChart" to 12

charts["captionTextColor"]

Summary:
Sets the textColor for the chart header.

Examples:
set the charts["captionTextColor"] of grp "myChart" to "red"
set the charts["captionTextColor"] of grp "myChart" to "127,80,0"

charts["captionAlign"]

Summary:
Sets the textAlign for the chart header.
Accepted entries are: left,right,center

Examples:
set the charts["captionAlign"] of grp "myChart" to "left"
set the charts["captionAlign"] of grp "myChart" to "right"
set the charts["captionAlign"] of grp "myChart" to "center"

charts["showValues"]

Summary:
Governs if values are rendered into each piece of the chart.

Examples:
set the charts["showValues"] of grp "myChart" to true
set the charts["showValues"] of grp "myChart" to false

charts["valuesNumberFormat"]

Summary:
Sets the numberFormat of the values being displayed for each piece of the chart.

Examples:
set the charts["valuesNumberFormat"] of grp "myChart" to "#.00"

charts["valuesTextFont"]

Summary:
Sets the font of the values rendered for each piece of the chart.

Examples:
set the charts["valuesTextFont"] of grp "myChart" to "Verdana"

charts["valuesTextSize"]

Summary:
Sets the textSize of the values rendered for each piece of the chart.

Examples:
set the charts["valuesTextSize"] of grp "myChart" to 12

charts["valuesTextColor"]

Summary:
Sets the textColor of the values rendered for each piece of the chart.

Examples:
set the charts["valuesTextColor"] of grp "myChart" to "red"
set the charts["valuesTextColor"] of grp "myChart" to "127,80,0"

charts["valuesAlign"]

Summary:
Sets the alignment of the value lable shown on each piece, relative to the center of the corresponding slice.
Values accepted are: left,right,top,bottom

Examples:
set the charts["valuesalign"] of grp "myChart" to "left"
set the charts["valuesalign"] of grp "myChart" to "right"
set the charts["valuesalign"] of grp "myChart" to "top"
set the charts["valuesalign"] of grp "myChart" to "bottom"

charts["valuesPrefix"] charts["valuesPostfix"]

Summary:
set the charts["valuesPrefix"] of grp "myChart" to "AB"

set the charts["valuesPostfix"] of grp "myChart" to " tons"

charts["showPoints"]

Summary:
Governs if a point for values is rendered into the chart.

Examples:
set the charts["showPoints"] of grp "myChart" to true
set the charts["showPoints"] of grp "myChart" to false

Comments:
This property does not have any effect if the charts["showValues"] is not set to true.

charts["pointsColor"]

Summary:
Sets the color for value points rendered into the chart.

Examples:
set the charts["pointsColor"] of grp "myChart" to "red"
set the charts["pointsColor"] of grp "myChart" to "127,80,0"

charts["pointsSize"]

Summary:
Sets the size for value points which are rendered into the chart.

Examples:
set the charts["pointsSize"] of grp "myChart" to 5

charts["pointsStyle"]

Summary:
Sets the style for value points which are rendered into the chart. Possible values are:
triangle, square, diamond, circle, <empty>

Examples:
set the charts["pointsStyle"] of grp "myChart" to "circle"
set the charts["pointsStyle"] of grp "myChart" to "triangle"
set the charts["pointsStyle"] of grp "myChart" to "square"
set the charts["pointsStyle"] of grp "myChart" to "diamond"

Comments:
Default value (if not set or set to empty) is "circle"