VISUAL ENGINEERING

KavaChart Developer Documentation

KavaChart PHP Users Guide

VERSION 5.0

KavaChart PHP Users Guide

164 Main Street • Second Floor • Los Altos, CA 94022
Phone 650.949.5410 • Fax 650.949.5578

Table of Contents

Chapter

1

KavaChart Introduction

This chapter provides an overview of how KavaChart works in a PHP scripting environment.

What is KavaChart?

KavaChart is a collection of tools for turning numbers into charts. KavaChart ProServe includes a charting engine that can create images and Macromedia Flash output "on-the-fly", based on dynamic data and style information.

KavaChart PHP is a very small bit of PHP code that gives you PHP scripting access to the KavaChart output engine. KavaChart PHP does not generate the chart image directly. Rather it creates a customized URL that includes encoded data and style information. This URL obtains a stream of image data, or Macromedia Flash, or SVG (Scalable Vector Grahics).

The KavaChart PHP chart object places this URL into the appropriate "IMG" or "OBJECT" tag, so that your PHP pages can include a mixture of HTML text and graphics. When your page is rendered on a browser, the browser makes an HTTP connection to the KavaChart image server to obtain the chart data. PHP is used only to construct the request.

By combining KavaChart's robust chart server with PHP's easy to use scripting techniques, you can add dynamic data-driven graphics to just about any web application with a minimum of programming.

The KavaChart Wizard

It's possible to design charts by hand, using a text editor and documentation about KavaChart's server object properties. It's a lot easier, though, to design your charts visually, using the KavaChart Wizard. This on-line tool provides a graphical interface for designing chart appearance, and the ability to combine your local data sources with chart designs. The Wizard produces complete output templates. These templates can be placed on the chart server to minimize the amount of information stored in the dynamic chart URL.

Chapter

2

Quick Start Guide

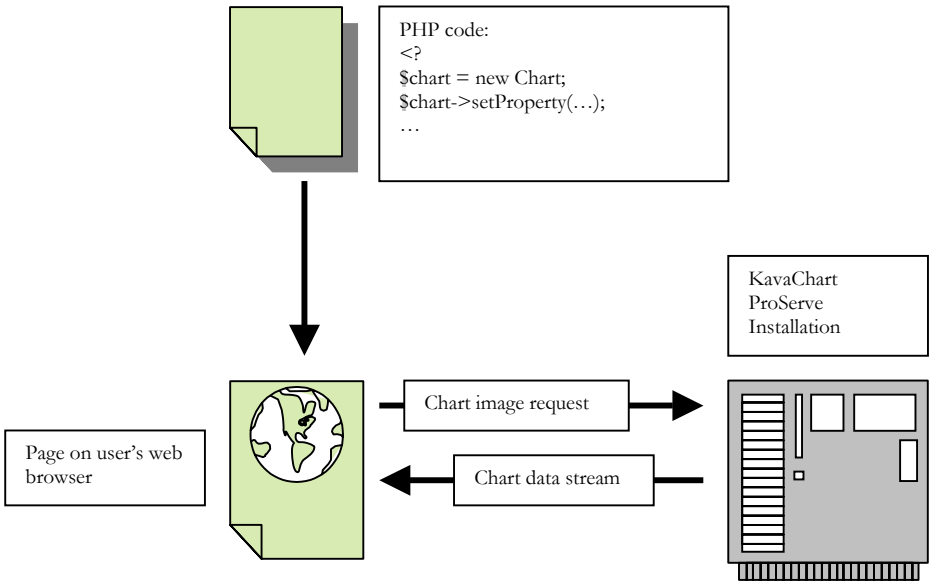
If you're the kind of user that wants to see results ASAP, follow this quick start guide to get KavaChart busy producing images right away.

Getting Started

Since KavaChart PHP can use a chart engine on a server other than your PHP-enabled server, to get started, you only need to place the kavachart.php file on your server. This file contains all the necessary logic to put KavaChart images on your pages.

The default server settings will use a Visual Engineering demo server, so your chart images will include a message like "KavaChart Images from ve.com". You can remove this message by installing a licensed version of KavaChart ProServe.

A typical chart session looks like this:



As usual, your PHP code is responsible for rendering dynamic content for the user's web page. Part of this content is an IMG or OBJECT tag that describes a chart to the KavaChart server.

As usual, the user's browser makes new HTTP connections to retrieve image and OBJECT information. The chart URL points to the KavaChart server, which builds and returns a chart image.

Sample Code

The PHP code to generate a simple chart is very brief:

```
<?
include 'kavachart.php';
$chart = new chart;
$chart->setProperty('dataset0yValues', '123,432,123,432');
?>
<html>
<body>
<p>
A chart:
<?=$chart->getImage()?>
</body>
</html>
```

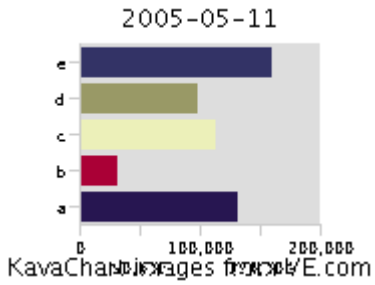
This little snippet of code produces a chart like this:

What really happened? We created a PHP "chart" object, and then set a "property" named "dataset0Yvalues" with the values "123,432,123,432". Whe our PHP code called "$chart->getImage()" the KavaChart PHP code produced an image tag that points to the VE demo server with all the property information, styles, chart type, and so on.

A chart's data, titles, colors, and other styles can be set using the chart object's "setProperty" method. Since you can use PHP variables for these properties, You can use PHP to build your overall chart. Here's an example:

```
<?
include 'kavachart.php';

$chart = new chart;

for($i=0;$i<5;$i++){
        $data = $data . (rand()/10000) . ',';
};

$chart->setProperty('dataset0yValues', $data);
$chart->setProperty('dataset0Labels', 'a,b,c,d,e');
$chart->setProperty('individualColors', 'true');
$chart->setProperty('titleString', date('Y-m-d'));

?>
<html>
<body>
<p>
<?=$chart->getImage()?>
</body>
</html>
```

This time we set some new properties, including some from PHP variables, to produce this chart image:



Again, the actual code is very simple. At this point you're probably wondering, "what are all these properties?". Charts can have a large number of properties, to define the most minor visual features (e.g. whether minor tick marks are displayed, or the line style for y axis grid lines) and some charts types have properties that only make sense for that chart type (e.g. startingAngle, for pie charts).

You can find complete documentation on the available chart properties in other chapters of this manual.

Chapter

3

Terminology Overview

It's helpful to understand KavaChart's terminology. Here's a visual description of some of the most basic terms:

KavaChart charts use a standard set of graphical and non-graphical components to do the work of representing your data. To get the most out of your charts, it's helpful to understand how KavaChart refers to these components and how they fit together.

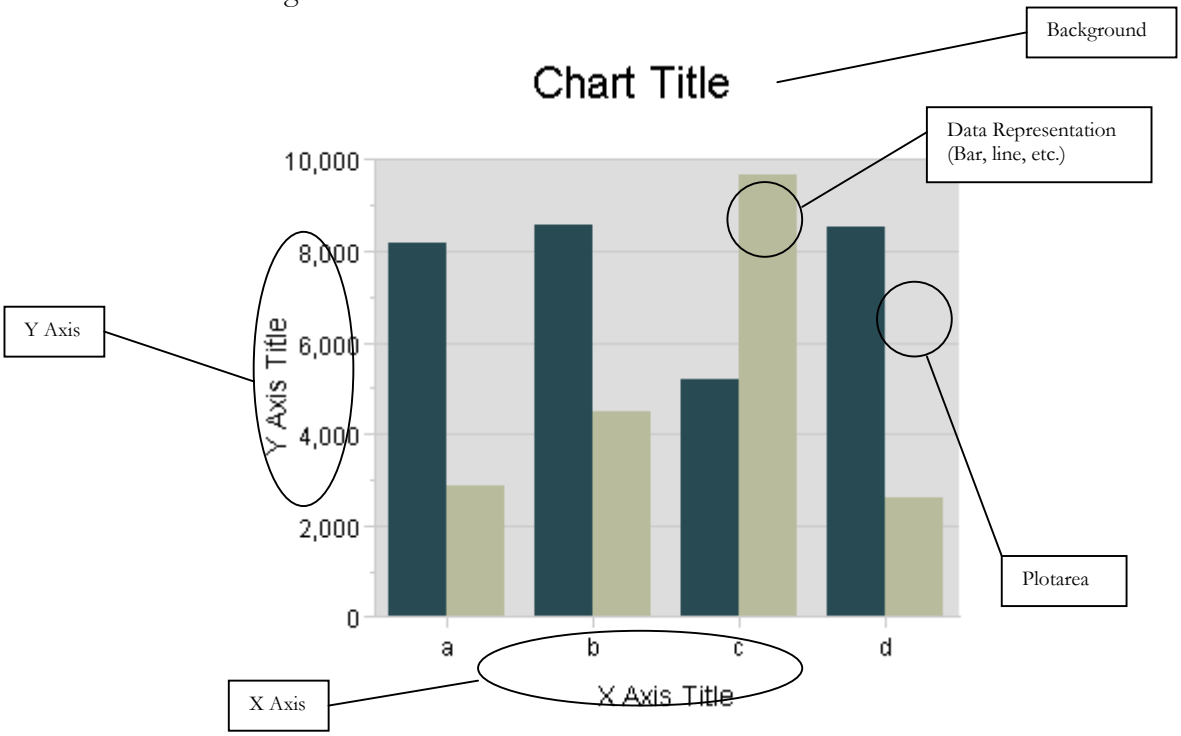
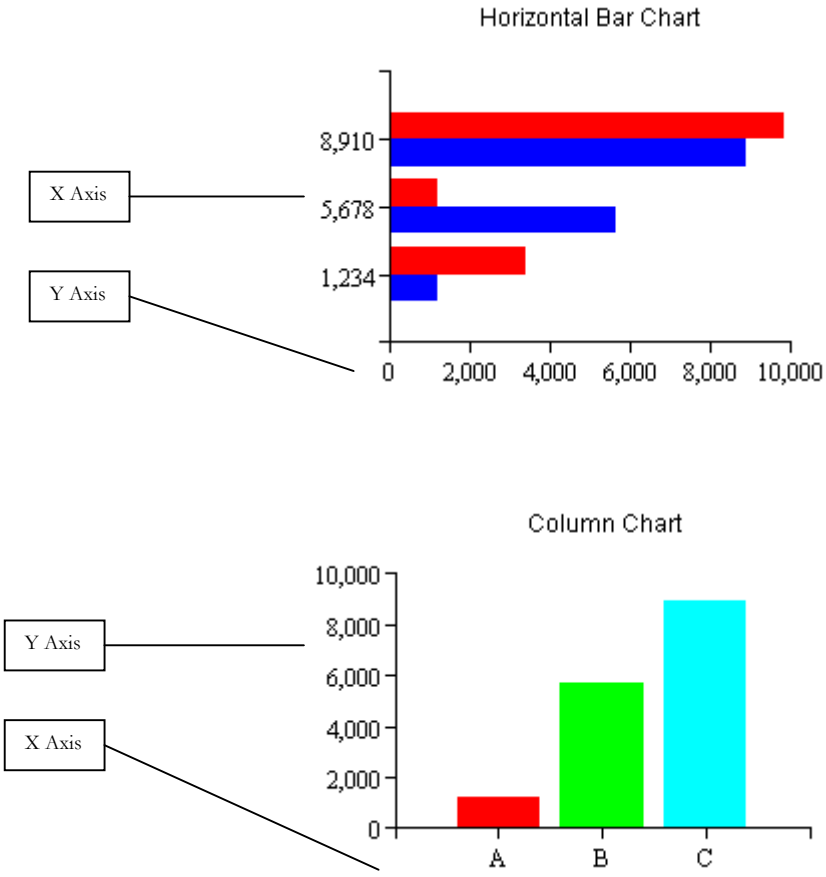


Chart Parts

X Axis and Y Axis



Axes can occur on the left, right, top or bottom of a Plotarea. A Y axis scales for Dataset Y values. Normally, these are represented vertically, and the Y axis is vertical. Horizontal Bar charts, Speedo charts, and Pie charts are exceptions.

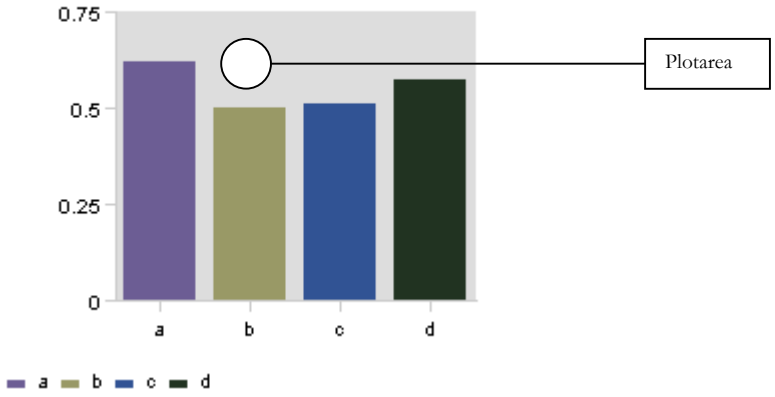
X axes scale for Dataset X values. For some charts, such as a Column chart or a Stacked Column chart, the X axis distributes the data evenly along the X axis, regardless of the Dataset's X values.

Several different types of Axes exist in KavaChart charts. A basic Axis automatically creates an aesthetically pleasing scale, arranged in even increments. An Axis can also scale logarithmically, which is appropriate for data with extremely wide variation. Some specialized axes, such as the DateAxis, are designed to handle specialized data. DateAxis arranges increments in months, weeks, or some other appropriate time value. A LabelAxis, such as those used for Column charts, will use user-defined labels. If no user-defined labels are present, the axis will try to determine appropriate labels.

Axes contain a number of elements that can be visible or not visible. These include the axis line, tick marks, minor tick marks, an axis title, labels, and grid lines. You can define the color of these elements, and in the case of labels and titles, the font. Labels can also use a number or date format of your choosing. By default, time and numeric labels are automatically localized for various locales.

Axes can be automatically scaled, semi-automatically scaled (you set the start and end, and let the axis determine labelling and increments), or manually scaled. A non auto-scaled axis requires you to set tick, grid, label, and minor tick counts as well as the axis start and end values.

Plotarea

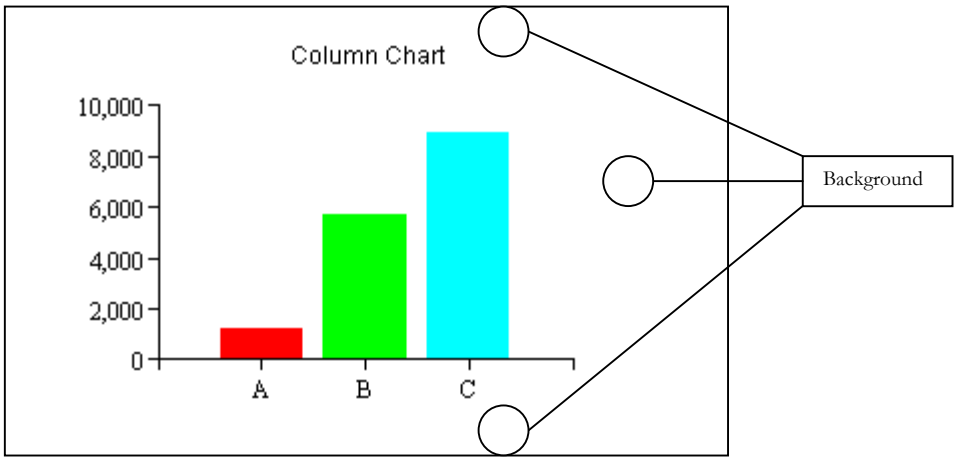


A Plotarea is the region bounded by an X and a Y Axis, which contains a DataRepresentation (such as a Line, Bar, Area, etc.). A Plotarea has a size and location determined by the upper right and lower left corners. The values that define the Plotarea size and location are percentages, relative to the overall chart. For example, an upper right corner value of (0.75, 0.75) means that the top of the Plotarea will be at 75% of the height, and the right side of the Plotarea will be at 75% of the width.

A Plotarea also has a user defined color and outline color.

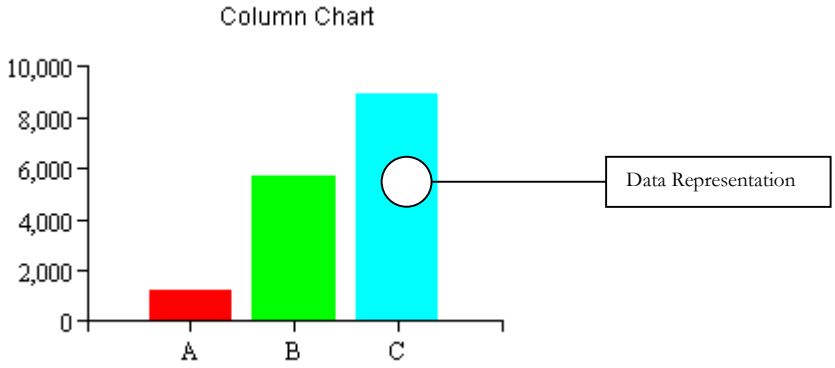
By changing the size and location of your Plotarea you implicitly change the size of your chart's margins. All Axis and DataRepresentation geometries will automatically adjust to accomodate your Plotarea definition.

Background



The rectangle underlying the entire chart is called a Background. The background also contains a title and sub-title. You can set the color of the background or use an image for the background if you prefer. You can also set the color and font of each of the title strings.

DataRepresentation

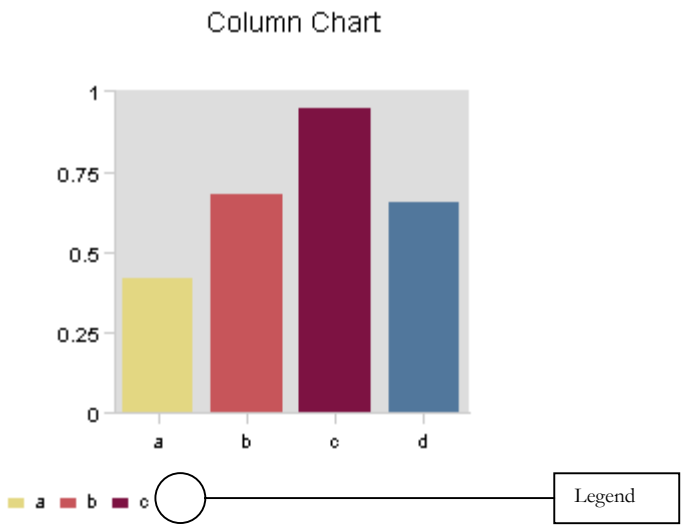


A DataRepresentation is the name KavaChart uses for a variety of objects. These include Line, Area, Bar, and Pie, as well as other more specialized DataRepresentations. These items visually describe a group of Datasets. For example, bar DataRepresentations exist that draw multiple series horizontally or vertically, and side by side or stacked. Bars also exist to represent high and low values, and to draw hi-lo-close, candlestick, histogram and other industry-specific visuals.

DataRepresentations obtain graphical information like colors and label fonts from the Datasets represented. Additionally, the X, Y (and other) magnitudes, as well as the bar/pie/etc. labels are derived from information in the Datasets.

Because DataRepresentations provide specific visual representations, they often have specialized properties. For example, bars can have variable cluster widths (the width of one group of bars), pies can vary the starting angle and toggle visibility on percent labels, speedos can have various types of needles, and so on.

Legend



A Legend contains a description of the Datasets in a particular chart. The icons and label text comes from the chart's Datasets. The X and Y values of a legend's lower left corner describe the legend's location. These values are in percentages of the overall chart. For example, a location of (0.5, 0.5) would place the lower left hand corner of the legend exactly at the center of your chart (50%, 50%).

Legends can have a background color, label font and font color. They can be arranged horizontally or vertically. You can also adjust the size of the legend's icons and the gap between the icon and the legend text (again, in percentages of the overall chart). Legends that are too large for the space you have defined will attempt to create a table of entries (rows and columns).

Various types of legends exist. These include standard Legends, which describe each dataset with a Dataset name and a rectangular icon, Pie legends, which describe each element in the first Dataset with an icon and the element name, and LineLegends, which use a line and optional marker for each Dataset.

Chapter

5

KavaChart PHP Object Functions and Chart Properties

This chapter discusses the functions and variables available in the KavaChart chart object, and the properties shared by all chart types.

Chart Object Functions

Generally, you can create a chart by creating an chart object instance, and then calling "setProperty" repeatedly to set up the chart attributes, and then calling "getImage()" to get the HTML code for image creation. Since the vast majority of chart object attributes are set by using property pairs, you will only use a few public methods with them. The public methods are listed below.

Method	Effect
$chart-> setProperty (name, value);	Sets a specified property to a specified value. Consult the list of server properties, general properties, and chart-specific properties for more information.
$chart->getImage();	Creates HTML code appropriate to producing a chart image with the data and styles you've defined.
$chart->setStyle(propertyFilename);	To use the chart engine most efficiently, you should place all non-dynamic style information into a properties file, and put that file on the chart engine server.
$chart->setChartType(chart type);	Names the charting application that should be used by the server. These are listed later in this document.
$chart->setResourceBundle(name);	If your application serves multiple locales, you can create a separate properties file for each locale. These files would contain localized titles, labels, and so on. The localized properties file is called a "resource bundle", and it has a "base name", such as "myStyles.properties". Localized versions are preceded by the 2 or 4 character locale specification. For example, "en_myStyles.properties" would be the English language version, and "es_myStyles.properties" would be the Spanish language version.

$chart->setLocale(abbreviation);	Sets the locale abbreviation to be used for loading a resource bundle.

Chart Object Variables

The KavaChart PHP object includes a few user variables that can be used to control how the object functions. Most importantly, these can be used to override the default server, and the default service used on that server.

Method	Effect
$chart-> imgHost	Sets the name of the host KavaChart server. The default is www.kavachart.com.
$chart->servletName	The name of the servlet used for the KavaChart ProServe engine. The default is "/KavaChartStream".

You can also override the way image URLs are constructed by modifying the variables $imgPrefix and $imgSuffix. Functions are provided to create Flash and SVG object containers. While these have been tested with all popular browsers, you may need to modify them for some applications.

Image Management Properties

The following table describes the built-in image management properties associated with the KavaChart server.

Property	value type	effect
imageType	String	Output image type. This defaults to PNG generation. Other supported imageTypes include: "flash" or "swf", which produces Macromedia Flash output, "svg", which produces Scalable Vector Graphics, "gif" , j_jpeg (a jpeg generator), j_png (a PNG generator, recommended), j_bmp (a .BMP file generator), j_ico (a .ICO file generator), j_xbm (an X-bitmap generator), j_xpm (X-pixmap generator).
height	integer	pixel height of generated image
width	integer	pixel width of generated image
antialiasOn	true/false	Turns antialiasing on for the resulting chart image.

Tooltip and Hyperlink Properties

If you use Macromedia Flash or Scalable Vector Graphics (SVG) image types, your charts can include tooltips and hyperlinks. This section describes the properties used to define these items.

Property	Value type	Effect
toolTips	true/false	Tells the object whether the chart should contain tooltips
hasLinkMap	true/false	Tells the object whether to use hyperlinks
dwellUseDatasetName	true/false	Tells the server whether to use the dataset name in the popup dwell labels
dwellUseLabelString	true/false	Tells the server whether to use each datapoint's label as a part of the popup dwell labels.
dwellUseXValue	true/false	Tells the server whether to use each datapoint's X value as a part of the popup dwell labels.
dwellUseYValue	true/false	Tells the server whether to use each datapoint's Y value as a part of the popup dwell labels.
dwellXString	String	A text string containing the characters "XX" to add descriptive text to the dwell label X value. Example: "Category XX"
dwellYString	String	A text string containing the characters "XX" to add descriptive text to the dwell label Y value. Example: "Unit Sales: $XX"
dwellXPercentFormat	true/false	Determines whether the X label will use a percent format
dwellYPercentFormat	true/false	Determines whether the Y label will use a percent format
dwellXCurrencyFormat	true/false	Determines whether the X label will use a localized currency format
dwellYCurrencyFormat	true/false	Determines whether the Y label will use a localized currency format
dwellXLabelPrecision	Integer	Number of digits of precision for dwell label values. For example, if precision is "2", labels will look like this: 123.45 or 123,45.
dwellYLabelPrecision	Integer	Number of digits of precision for dwell label values. For example, if precision is "2", labels will look like this: 123.45

Hyperlinks

KavaChart can generate hyperlinks in Macromedia Flash and SVG output. This is useful if you want to create "drill-down" reports keyed to individual data items. For example, a pie chart might represent revenues in each district of a county. Clicking on a slice could take you to another chart that describes the revenue breakdown for that individual district.

Hyperlinks work the same as tooltips, but use some additional properties:

Property Name	Value	Description

datasetNLinks	List	A list of comma-separated URLs that will be used as hyperlinks for each Datum (bar, pie slice, line marker, etc.) in Dataset N.

Data Related Properties

Every chart creates a graphical representation of numeric information. Different kinds of charts require different kinds of numeric information, but every chart requires at least some sort of numbers to start with. KavaChart organizes this information into "Datasets", which contain the numbers and text required by your chart.

Some charts have a single dataset (pie charts and speedos), while others may have many datasets (each line on a line chart is a different dataset). Similarly, some datasets contain a lot of information for each observation (a candlestick chart has a time, high, low, open, close, and label value for each price bar), while others contain only a little (a speedo uses only a single value, and a pie uses one value and one label for each slice).

Following mathematical conventions, the most basic numeric unit for each observation in a chart is called a "Y" value. This means that we use "Y" values to define the value for each slice in a pie, or the height of each column, or even the width of a bar in a horizontal bar chart. Y values are required for any chart to create a meaningful visual.

Every chart can also contain a textual label for each Y value. These charts don't always display that label, but it's available. For example, you might assign some labels like "East", "West", "North", and "South" to a bar chart. The labels might not be visible on the chart, but you could use them in a tooltip label for users that want to explore further.

Some charts also use "X" values, which is generally thought of as the "independent", or deterministic part of your observation. For example, if your chart shows how ozone levels compare to temperature, you would assume that temperatures are "independent" of ozone levels, while ozone levels may be "dependent" on temperatures. Temperature would be used as "X" values in this case. A line chart that plots ozone levels against temperature might have a variety of temperature observations that don't fall into neatly defined categories, but for each temperature observed (X), there would also be an ozone level observation (Y).

Not all charts use X values. In some cases (pie charts) this is obvious. In other cases, it may not be. For example, bar charts don't usually use X values, because

bars are generally used to represent categories, rather than a set of independent numeric values. In the case of a bar or column chart, KavaChart will ignore your X values, and use a set of implied X values (0, 1, 2, ...).

More complex charts, such as hi-lo bar charts or financial charts (OHLC, Candlestick) require additional information, which we call "Y2" and "Y3" data. This auxiliary information takes on special meaning depending on the chart that calls for it.

All this X, Y, Y2, and Y3 data is organized into datasets. Every chart can contain up to 40 datasets, with an unlimited number of observations in each dataset. Some charts (speedo and pie, for example) don't use all the data; these charts use the lowest numbered information available. For example, pie charts use dataset 0, and speedos use only observation 0 of dataset 0.

In addition to the numbers and text, each observation can also take a fill color definition, a line color, and a fill style and line style. The dataset that contains the observations also has fill, line, and color information, and a name for the overall dataset. Different charts use all this information in different ways.

For example, a pie chart uses the color definitions for each observation to draw each slice, and individually colored bar charts use this information for each bar's color and for legend icons. Standard bar charts and line charts use the dataset colors and labels for drawing and legends.

Dataset Properties

The table below gives parameter names and usage descriptions. All parameters listed as "dataset0" are valid for datasets 0 through 39. Items described as "lists" expect a comma separated list of values, colors, etc. You can change the delimiter from a comma to another character with the "delimiter" param.

Property Name	Type	Effect
dataset0xValues	list	comma separated list of X values for dataset 0.
dataset0yValues	list	comma separated list of Y values for dataset 0
dataset0y2Values	list	comma separated list of difference values for dataset 0 hilo bars
dataset0xyValues	List	comma separated list of X,Y values for dataset 0.
dataset0dateValues	List	Comma separated list of time/date strings for dataset 0. See also "inputDateFormat".
dataset0y3Values	list	Tertiary observations for charts that require 3 Y values (e.g. hi-lo-close charts)

Discontinuities

One special case deserves notice here. Some charts support the notion of "discontinuities" (disLineApp, disDateLineApp, etc.). In these charts, you want to have a break in the line or some other visual feedback that shows missing data. In this case, you can just use some non-number, like 'x', to indicate a break. KavaChart recognizes this as a missing point and creates the line break as appropriate. Here's an example:

```
<?
$chart = new chart;

//add some data here:
$chart->setProperty("dataset0yValues","2,3,6,x,4,5,2,x,7,8");
?>
<p>
Here's the chart:
<br>
<?=$chart->getImage()?>
```

Time oriented charts

Charts that display time oriented data (dateLineApp, dateAreaApp, etc.) use time stamps as a special kind of numeric value. For these charts, use the property dataset0dateValues, like this:

```
$chart = new chart;
$chart->setChartType("dateLineApp");
$chart->setProperty("width", "300");
$chart->setProperty("height", "200");
$chart->setProperty("dataset0yValues", "234,321,234");
$chart->setProperty("dataset0dateValues",
                    "01/01/02,02/01/02,03/01/02");
```

This property translates the dates into a form usable by the KavaChart server and places our Y observations at the proper locations along the axis. Unfortunately, our date definitions are ambiguous here. Did our observations occur on January 1, 2, and 3? Or did they occur on January 1, February 1, and March 1?

Managing Date Formats

To properly use dataset0dateValues, you should also use inputDateFormat:

```
$chart = new chart;
$chart->setChartType("dateLineApp");
$chart->setProperty("width", "300");
$chart->setProperty("height", "200");
$chart->setProperty("inputDateFormat", " MM/dd/yy");
$chart->setProperty("dataset0yValues", "234,321,234");
$chart->setProperty("dataset0dateValues",
                    "01/01/02,02/01/02,03/01/02");
```

The table below describes how to construct an inputDateFormat to match your data generator.

Field	Full Form	Short Form
Year	yyyy (4 digits)	yy (2 digits)

Month	MMM (name)	MM (2 digits), M (1 or 2 digits)
Day of week	EEEE	EE
Day of Month	dd (2 digits)	d (1 or 2 digits)
Hour (1-12)	hh (2 digits)	h (1 or 2 digits)
Hour (0-23)	HH (2 digits)	H (1 or 2 digits)
Hour (0-11)	kk (2 digits)	k (1 or 2 digits)
Hour (1-24)	KK (2 digits)	K (1 or 2 digits)
Minute	mm	None
Second	ss	None
Millisecond	SSS	None
AM/PM	a	None
Time Zone	zzzz	zz
Day of Week in Month	F (e.g. 2nd Tuesday)	None
Day in year	DDD (3 digits)	D (1, 2, or 3 digits)
Era	G (e.g. BC or AD)	None

Time and date oriented charts have special properties for managing axes, which are listed below.

Color and Style Properties

KavaChart server charts support a lengthy list of properties to help you make your charts look exactly the way you want. These properties are used to set colors, fonts, textures, line styles, and the overall layout of your chart.

Color and style propertes take different kinds of values. The table below gives you some examples of what these values should be. Also, all properties are case sensitive.

Parameter Type	Explanation	Example

Integer	An integer value, like "1", or "7". This is usually used to specify something like a line style or a marker style; one out of a list of several available types.	`$chart->setProperty("legendTexture","1");`
		
		`legendTexture=1`

Double	A real number value, like 0.25. Generally, these values are expressed in terms of a percentage of the	`$chart->setProperty("plotAreaBottom","0.12");`
		
		`plotAreaBottom=0.12`

	overall chart size.	
Font	font parameters include information for the font name, the font size, and the font style. The example instructs KavaChart to use 18 point Arial italic fonts for this chart's X axis labels. 0 is plain, 1 bold, and 2 italic.	`$chart->setProperty("xAxisLabelFont","Arial,18,2");`
Color	this field expects a color name, or a hexadecimal color definition (in RGB). Valid colors names in these charts include black, white, gray, darkGray, lightGray, red, pink, orange, yellow, green, magenta, cyan, and blue. A valid hex definition for white is "ffffff". You can also use the color "transparent" if you don't want a particular element to be visible.	`$chart->setProperty("titleColor","ffbb00");`
List	These fields are looking for a list of items, separated by a delimiter. The default delimiter is a comma character, but you can change this with the "delimiter" param.	`$chart->setProperty("dataset0Colors","green,red,ff00aa">`
String	A text string	`$chart->setProperty("titleString", "hello, world");`
url	These fields expect some URL available to the KavaChart server	`$chart->setProperty("backgroundImage","/tmp/pic.jpg");`
Boolean	Either "true" or "false"	`$chart->setProperty("outlineLegend","false");`
Anything	Some parameters can take any value. The server just wants to know if the parameter has been defined	`$chart->setProperty("3D","yeah, sure");`

General Color and Font Properties

The first group of properties apply to all chart types. These properties define colors, overall layout, titles, and so on.

Parameter	Value Type	Effect

colorPalette	String	Set the overall default color palette for the chart. Default possibilities: web_sanfrancisco, web_minnesota, web_alaska, web_newyork, web_losangeles, web_grays, web_seattle, web_newmexico, web_rosemary, web_pastel, web_prague, presentation_cool, presentation_browns, presentation_southwest, presentation_impact, presentation_deep, presentation_oceana, presentation_sophisticated The default is “web_newyork”
colorPaletteDefinition	list	List of color definitions (e.g. 00ff00,green,blue,black)
titleString	String	Chart Title (default none)
titleFont	font	Font name, size, & style for chart title (default TimesRoman, plain, 12 pt)
titleColor	color	color of text in Title (default black)
titleX	double	X location of the title string, if this is not specified the title will be centered.
titleY	double	Y location of the title string.
subTitleString	String	Chart Sub-Title (default none)
subTitleFont	font	Font name, size, & style for chart title (default TimesRoman, plain, 12 pt)
subTitleColor	color	color of text in Title (default black)
subTitleX	double	X location of the subtitle string, if this is not specified the subtitle will be centered.
subTitleY	double	Y location of the subtitle string.
labelsOn	anything	determines whether bar, line, pie, etc., labels will be visible
labelAngle	integer	the number of degrees to rotate datum labels
labelPrecision	integer	the number of digits of precision for datum labels
legendOn	anything	make the legend visible
legendOff	anything	make the legend invisible (default)
legendColor	color	sets the background color of a legend
legendVertical	anything	legend icons in vertical list (default)
legendHorizontal	anything	legend icons in horizontal list
legendLabelFont	font	Font name, size, & style for legend (default TimesRoman, plain, 12 pt)
legendLabelColor	color	color of text in legend (default black)

legendllX	double	X location of lower left legend corner (default 0.2)
legendllY	double	Y location of lower left legend corner (default 0.2)
iconWidth	double	width of legend icon (default 0.07)
iconHeight	double	height of legend icon (default 0.05)
iconGap	double	gap between icon and next legend entry (default 0.01)
legendSecondaryColor	color	The Color to be used as the secondary color for this legends texture/gradient.
legendGradient	integer	Sets the gradient for this legend. Available gradient values are 0 for left/right mirrored, 1 for top/bottom mirrored, 2 for top to bottom, and 3 for left to right
legendTexture	integer	Sets the texture for this legend. Available texture values are 0 for horizontal stripes, 1 for vertical stripes, 2 for diagonal down stripes, 3 for diagonal up stripes, 4 for cross hashing, and -1 to use the legendimage to create the texture.
legendImage	URL (or filename)	image to use for this legend's background (default none). Use this property to define line markers for scatter plots.
legendLineWidth	integer	pixel width of legend outline
legendLineStyle	integer	Sets the line style for this legend's outline. Available values for this parameter are 0 for dashed, 1 for dotted, 2 for dot-dashed, and -1 for solid (default = -1).
plotAreaTop	double	top of the plotting area
plotAreaBottom	double	bottom of the plotting area
plotAreaRight	double	right side of the plotting area
plotAreaLeft	double	left side of the plotting area
plotAreaColor	color	color of plotting area background (default white)
plotAreaSecondaryColor	color	The Color to be used as the secondary color for this plotarea's texture/gradient.
plotAreaGradient	integer	Sets the gradient for this plotarea. Available gradient values are 0 for left/right mirrored, 1 for top/bottom mirrored, 2 for top to bottom, and 3 for left to right
plotAreaTexture	integer	Sets the texture for this plotarea. Available texture values are 0 for horizontal stripes, 1 for vertical stripes, 2 for diagonal down stripes, 3 for diagonal up stripes, 4 for cross hashing, and -1 to use the plotarea image to create the texture.
plotAreaImage	URL (or filename)	image to use for this plotarea's background (default none). Use this property to define line markers for scatter plots.
plotAreaLineWidth	integer	pixel width of plotarea outline
plotAreaLineStyle	integer	Sets the line style for this plotarea's outline. Available values for this parameter are 0 for dashed, 1 for dotted, 2 for dot-dashed, and -1 for solid (default = -1).
backgroundColor	color	color of chart background (default white)
backgroundSecondaryColor	color	The Color to be used as the secondary color for this background's texture/gradient.

backgroundGradient	integer	Sets the gradient for this background. Available gradient values are 0 for left/right mirrored, 1 for top/bottom mirrored, 2 for top to bottom, and 3 for left to right
backgroundTexture	integer	Sets the texture for this background. Available texture values are 0 for horizontal stripes, 1 for vertical stripes, 2 for diagonal down stripes, 3 for diagonal up stripes, 4 for cross hashing, and -1 to use the plotarea image to create the texture.
backgroundImage	URL (or filename)	image to use for this background's background (default none).
backgroundLineWidth	integer	pixel width of background outline
backgroundLineStyle	integer	Sets the line style for this background's outline. Available values for this parameter are 0 for dashed, 1 for dotted, 2 for dot-dashed, and -1 for solid (default = -1).
3D	anything	turns on 3D effects for this chart (default 2D)
2D	anything	turns on 2D effects for this chart (default 2D)
XDepth	integer	number of pixels of offset in X direction for 3D effect (default 15)
YDepth	integer	number of pixels of offset in y direction for 3D effect (default 15)
delimiter	String	the separator character for list parameters. Default is comma (e.g. "123.432.123").
outlineColor	Color	Color to use for outlining bars, plotareas, etc. (Default none). Using this param automatically enables outlining for most objects
outlineDataRepresentation	true/false	If outlineColor is set to some color, you can selectively turn the outlining off for the DataRepresentation (Bars, Pie, Area, etc.) by setting this property to "false". Default is "true".
outlinePlotarea	true/false	If outlineColor is set to some color, you can selectively turn the outlining off for the Plotarea (the region bounded by the x and y axes) by setting this property to "false". Default is "true".
outlineBackground	true/false	If outlineColor is set to some color, you can selectively turn the outlining off for the Background (the total chart image area) by setting this property to "false". Default is "true".
outlineLegend	true/false	If outlineColor is set to some color, you can selectively turn the outlining off for the chart Legend by setting this property to "false". Default is "true".
showVersion	true/false	If this is set to true, the chart will be created with the version number replacing the chart's title.
annotation0LabelString	String	A label for note 0 (unlimited notes available) *Note: a "|" character will break this note into multiple lines.*
annotation0Alignment	above|below|left|right	Where note should appear relative to location
annotation0CoordinateSpace	pixel|axis	Coordinate space for location values
annotation0Xloc	Number	Pixels or axis values
annotation0YLoc	Number	Pixels or axis values

annotation0LabelFont	Font	Font for this note
annotation0LabelColor	Color	Font color for this note
annotation0FillBackground	true|false	Determines whether this note will have an opaque background
annotation0BackgroundColor	Color	Note's background color
annotation0OutlineColor	Color	This note's outline color (if any)

Axis Related Properties

The following tables contain properties for adjusting axes. Line, area, bar, and their derivatives use these properties. Axis properties include individual properties and an option list. The option list groups several properties together.

Axis Option Lists

The option lists include various options for adjusting the look of an X or Y axis. Use these parameters in a list, like this:

```
setProperty("xAxisOptions" "gridOff,tickOff,lineOn");
```

If you're modifying an auxiliary Y axis (such as in a chart that has left and right axes), use the property name auxAxisOptions.

yAxisOptions (xAxisOptions)	Effect
autoScale	automatically create axis scale (default)
noAutoScale	axis scale defined in other properties
rotateTitle	"true" if vertical axis title should be parallel with axis
logScaling	"true" if axis should use log scaling
lineOn	axis line is visible (default)
lineOff	axis line is invisible
tickOn	major tick marks are visible (default)
tickOff	major tick marks are invisible
minTickOn	minor tick marks are visible
minTickOff	minor tick marks are invisible (default)
labelsOn	axis labels are visible (default)
labelsOff	axis labels are invisible
gridOn	grid lines are visible
gridOff	grid lines are invisible (default)
rightAxis	this axis goes on the right
topAxis	this axis goes on the top
bottomAxis	this axis goes on the bottom
leftAxis	this axis goes on the left (default)
percentLabels	this axis should use localized percentage representations (not valid for date and label axes)

currencyLabels	This axis will use a localized currency representation for labels (not valid for date and label axes)

Detailed Axis Properties

If you're modifying an X Axis (usually on the top or bottom of a chart), use xAxis*PropertyName* instead of yAxis*PropertyName*. X Axes are on the left and right for Horizontal Bar Type charts. Speedo and Polar charts have a single Axis, which is a Y Axis.

If you're modifying an Auxiliary Y Axis (charts that have left and right axes, for example), use auxAxis*PropertyName* instead of yAxis*PropertyName*.

Axis Property	Value Type	Effect
yAxisTitle	string	Axis title
yAxisTitleFont	font	Axis title font
yAxisTitleColor	color	Axis title color
yAxisLabelFont	font	use this font for axis labels
yAxisLabelColor	color	axis labels in this color (default black)
xAxisLabels	list	A comma separated list of user-defined labels for this Axis. This is only effective for certain types of chart (BarChart derivatives, LabelLineChart, Area charts) that use a LabelAxis. By default, LabelAxis is only used for X axes.
yAxisLabelAngle	integer	label rotation in degrees (default 0). Note: rotations of 0 and 90 degrees will be the most readable
yAxisLabelPrecision	integer	Number of digits past the decimal point to display
yAxisLineColor	color	axis line color (default black)
yAxisTickColor	color	axis tick mark color (default black)
yAxisGridColor	color	axis grid line color (default black)
yAxisColor	color	sets axis grids, ticks, lines and labels to the same color
yAxisTickLength	integer	number of pixels long for axis tick marks
yAxisMinTickLength	integer	number of pixels long for axis minor tick marks
yAxisStart	double	starting value on axis. By default, axes automatically determine a starting and ending value. By setting this value, you can give the axis a default minimum value. If the Axis is set to noAutoScale, this value will be used directly. Otherwise, this value may be adjusted slightly to yield better looking labels. For example, if you set yAxisStart to 0.01, the chart may decide to round the value down to 0.0 to create even axis increments.
yAxisEnd	double	ending value on axis. By default, axes automatically determine a starting and ending value. By setting this value, you can give the axis a default maximum value. If the Axis is set to noAutoScale, this value will be used directly

		Otherwise, this value may be adjusted slightly to yield better looking labels. For example, if you set yAxisStart to 9.99, the chart may decide to round the value up to 10.0 to create even axis increments.
yAxisLabelCount	integer	how many labels on an axis set to noAutoScale
yAxisTickCount	integer	how many tick marks on an axis set to noAutoScale
yAxisMinTickCount	integer	how many minor tick marks on an axis set to noAutoScale
yAxisGridCount	integer	how many grid lines on an axis set to noAutoScale
yAxisGridStyle	integer	the line style of the grid lines for this axis
yAxisGridWidth	integer	the width in pixels of the grid lines for this axis
yaxisThresholdLine0Color	Color	The color of reference line 0 (40 available).
yAxisThresholdLine0LabelColor	Color	The color of the label for reference line 0
yAxisThresholdLine0LabelFont	Font	Font for for reference line 0's label
yaxisThresholdLine0LabelString	String	Optional label for reference line 0
yAxisThresholdLine0LineStyle	Integer	Line style for reference line 0
yAxisThresholdLine0Value	Double	Where on the Y axis should reference line 0 draw.

Tip:

If you want an axis to start at a specific value, but end at some value based on data, just use yAxisStart without including noAutoScale among your yAxisOptions.

Date and Time Axis Properties

The following list contains options for Time/Date X axes, such as those used for dateLineApp and dateAreaApp, as well as financial chart types like stickApp and hiLoCloseApp

DateAxis Properties	Type	Effect
startDate	string	time/date for axis starting value.
endDate	string	time/date for axis ending value
axisDateFormat	string	By default, DateAxis selects an appropriate labelling type based on your time range and your locale. This property lets you override the axis labels to use your specific formatting instructions. See the Date Format table above for more information on how to use the formatting patterns.
axisSecondaryDateFormat	string	Some DateAxes use a primary and secondary format to highlight important boundaries, like years or hours. This parameter lets you set the date or timestamp format for one of these boundaries. See the Date Format section above for more information on how to use the formatting patterns.

scalingType	integer	1 scale by seconds 2 scale by minutes 3 scale by hours 4 scale by days 5 scale by weeks 6 scale by months 7 scale by years
axisTimeZone	string	This determines the timezone used for displaying date data. By default chart objects use the timezone of the server. This may be incorrect in some cases. For example, if my server is parsing time data in New York, and I want a user in California to see the data in real-time not New York time, then this parameter can be used to change the way the data is displayed. Timezones can be specified by JDK 1.1 deprecated strings like PST, EST, etc., by Java 2 standards: "America/Los_Angeles", or by the difference from GMT in this syntax: GMT[+|-]hh[[:]mm] (for example Eastern Standard Time would be equivalent to "GMT-5:00").
inputTimeZone	string	This determines the timezone used for parsing date data. By default chart objects use the timezone of the . Timezones can be specified by JDK 1.1 deprecated strings like PST, EST, etc., by Java 2 standards: "America/Los_Angeles", or by the difference from GMT in this syntax: GMT[+|-]hh[[:]mm] (for example Eastern Standard Time would be equivalent to "GMT-5:00").

Dataset Related Color and Style Parameters

Dataset colors and styles are very important to KavaChart. These colors are used to define the color of bars, pie slices, legend icons, and so on.

Dataset Parameters (available datasets 0 through 39)	Type	Effect
dataset0Name	string	name for display in legend (default "dataset0")
dataset0Color	color	color to use for this dataset (default varies)
dataset0Colors	list of colors	colors to use for pie slices or bars (default varies)
dataset0SecondaryColor	color	The Color to be used as the second color with dataset textures/gradients. The default is transparent.
dataset0SecondaryColors	list of colors	Colors to be used as the second color with dataset textures/gradients. The default is transparent.
dataset0Gradient	integer	Sets the gradient for this dataset. Available gradient values are 0 for left/right mirrored, 1 for top/bottom mirrored, 2 for top to bottom, and 3 for left to right
dataset0Gradients	list of integers	Sets the gradients for this dataset. For available values see datset0Gradient.
dataset0Texture	integer	Sets the texture for this dataset. Available texture values are 0 for horizontal stripes, 1 for vertical stripes, 2 for diagonal down stripes, 3 for diagonal up stripes, 4 for cross hashing, and -1 to use the dataset image to create the texture.
dataset0Textures	list of integers	Sets the textures for this dataset. For available values see datset0Texture.

dataset0Image	Filename or URL	image to use for this dataset's markers (default none). Use this property to define line markers for scatter plots.
dataset0Images	list of filenames	images to use for this chart's markers (default none). Use this property to define individual line markers for scatter plots. These values will also be used as fill images for pie charts or individually colored bar charts.
dataset0MarkerStyle	integer	Specify an internal marker for line charts and scatter plots (0=box, 1=diamond, 2=circle, 3=triangle). Default is -1 (none)
dataset0MarkerStyles	list of integers	Specify internal markers for datsets drawn with different markers at each data point. See datset0MarkerStyle for available marker values.
dataset0MarkerSize	integer	pixel width of internal marker for line charts and scatter plots.
dataset0MarkerSizes	list of integers	pixel widths of internal markers for line charts with individual markers
dataset0LineWidth	integer	pixel width of plot line
dataset0LineStyle	integer	Sets the line style for this line. Available values for this parameter are 0 for dashed, 1 for dotted, 2 for dot-dashed, and -1 for solid (default = -1).
dataset0LabelFont	font	font to use for this dataset's labels (default TimesRoman 12pt)
dataset0LabelColor	color	color to use for this dataset's labels (default black)

Chapter

6

Server Chart Objects

This chapter details the specific chart objects available in KavaChart ProServe. These charts are the types you name with the setChartType function.

Each chart type has a few properties that deal with the specifics of that chart type. For example, pie charts have a property that lets you set the starting angle of the pie. This property doesn't make sense for bar charts.

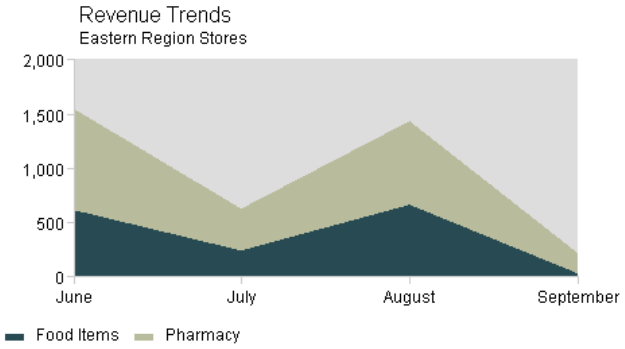
These charts are specified by using the "setChartType" function. For example, if you wanted to use a column chart, you'd use code like this:

```
<?
$chart = new chart;
$chart.setChartType('columnApp');
?>
```

Area Charts

An area chart uses polygons to describe trends. This type of chart is most appropriate for trends that include cumulative values. For example, an area chart may be most appropriate for displaying revenue trends for several categories. The overall trend appears at the top, while each item's contribution would appear as a layer.

areaApp

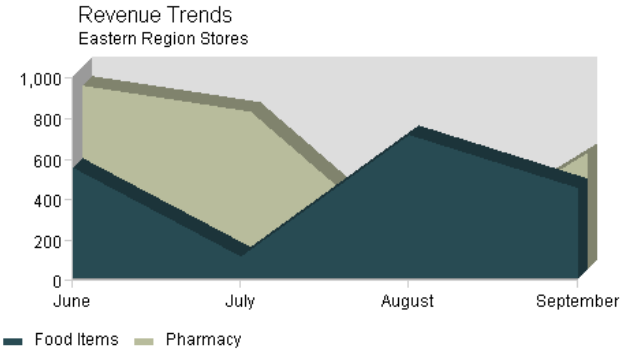


AreaApp ignores your X value specifications and assumes the values are 0, 1, 2, 3, … This ensures that the areas will align properly. Use the xAxisLabels parameter to specify your actual labels.

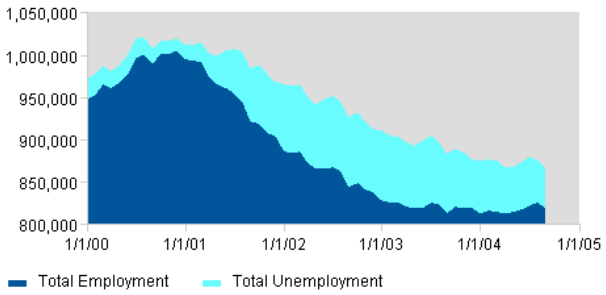
Because area charts are used to display cumulative trends, they don't generally give a clear idea of where individual data points are. For this reason, they're most appropriate for general trends. Also, dwell labels and hyperlink hot spots run from mid-point to mid-point for this type of chart.

It's important for area charts with multiple datasets to use the same X values for every dataset. Otherwise the areas cannot stack properly.

Note that un-stacked, 3D area charts are problematic. Areas can become completely obscured, as in the final observation in the chart below:



dateAreaApp



AreaApp ignores your X value specifications and assumes the values are 0, 1, 2, 3, … This ensures that the areas will align properly. Use the xAxisLabels property to specify your actual labels.

DateAreaApp assumes that X values are timestamps as described in the section above on data for time oriented charts. It also uses X axis parameters for time oriented charts.

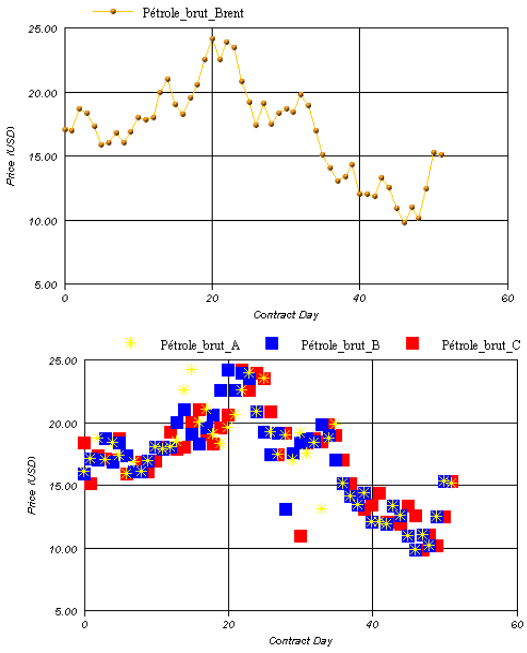
Because area charts are used to display cumulative trends, they don't generally give a clear idea of where individual data points are. For these charts, "getLinkMap()" define regions that go from mid-point to mid-point for each observation.

Property	value type	effect
baseline	double	sets the baseline value for this area
stackAreas	true/false	determines whether the areas will be stacked on top of each other (default is true)

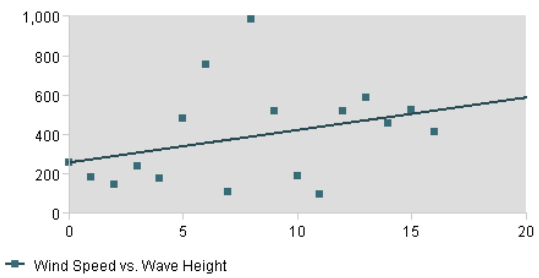
Line and Scatter Charts

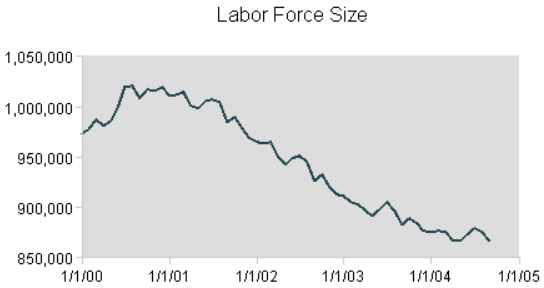
These charts include:

lineApp

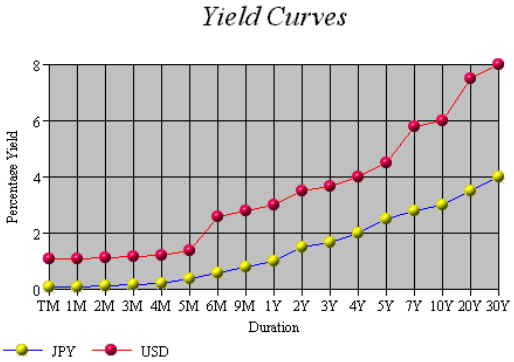


regressApp



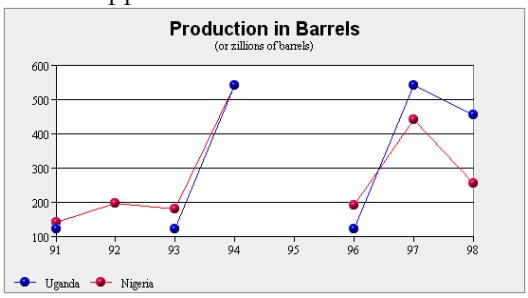


dateLineApp



labelLineApp

disLineApp



disLabelLineApp

In general, these charts can be used as conventional line charts, with or without markers at each vertex. Plot lines can be turned off with the "plotLinesOff" property or the dataset0Color property. If markers are turned on with lines turned off, these charts become scatter plots. You can plot some dataset lines and make others invisible by setting dataset0Color to "transparent" for the scatter-only datasets.

Chart objects that begin with "dis", such as disLineApp, support discontinuous data. They will create line breaks where data is missing. See the data section above to understand how to define discontinuities

DateLineApp uses time oriented data, as discussed in the data section above. These charts also recognize properties for formatting time oriented axes, discussed above.

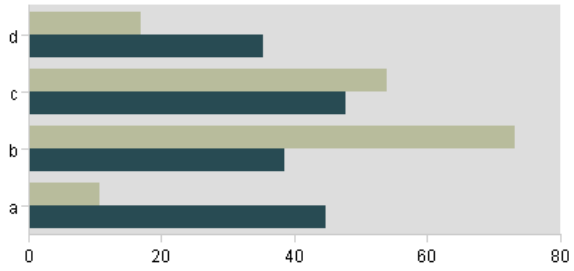
RegressApp performs a simple linear regression calculation on the chart's data values. Markers appear at the actual data points, while the line is drawn according to the regression's prediction. This is a classic "scatter plot" that shows positive, negative, or no correlation, and gives visual feedback about the strength of that correlation.

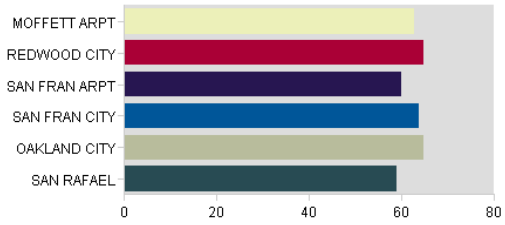
Property	value type	effect
plotLinesOn	anything	plot lines should display (default)
plotLinesOff	anything	Create a scatter plot by making plot lines invisible
individualMarkers	true/false	If markers are used, this parameter determines whether or not the datum markers will be used rather than the dataset marker (default is false).

Bar and Column Charts

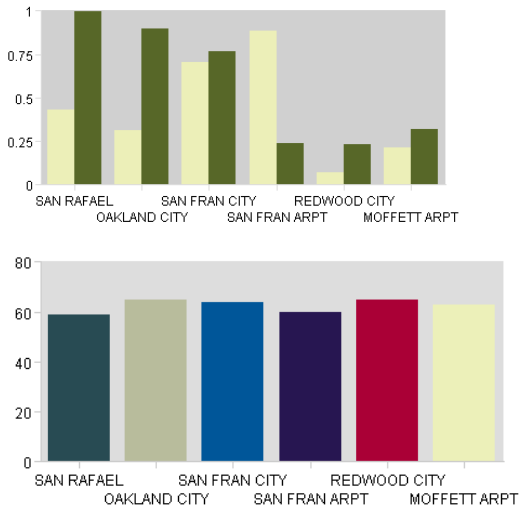
This category includes both charts with vertical and horizontal bars, as well as hi-lo bars. The charts are:

barApp

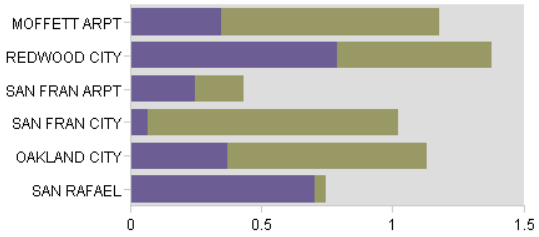




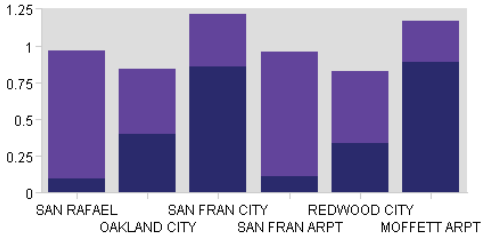
columnApp



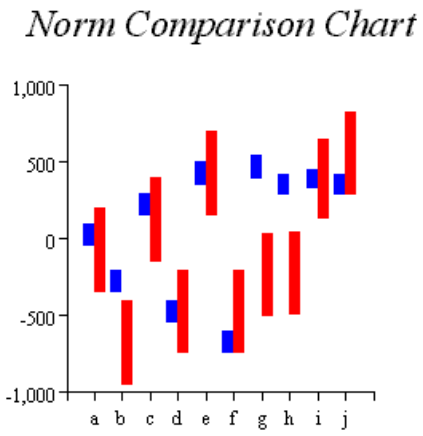
stackBarApp



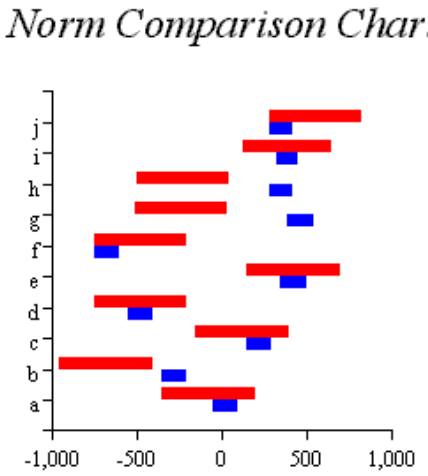
stackColumnApp



hiLoBarApp



hHiLoApp (horizontal hi lo bars)

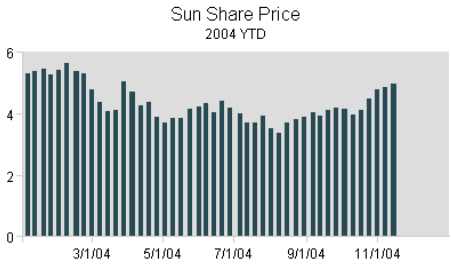


dateBarApp

dateColumnApp

dateStackBarApp

dateStackColumnApp



Bar charts have variable bar width, an adjustable baseline, and labels that can be toggled on or off. If you don't include a parameter to define X axis labels, this chart will use item labels (parameter dataset0Labels) beneath each bar. If item labels aren't defined, this chart will display each bar's Y value beneath it.

If you want each bar to have a different color, set the property "individualColors" to true, and define the colors with "dataset0Colors".

StackBarApp and stackColumnApp stack datasets instead of clustering them. This is useful to display a cumulative summary along with the individual data.

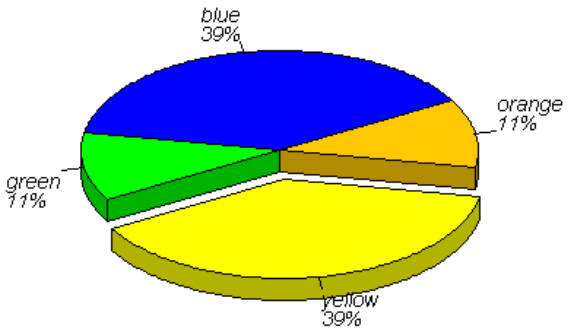
Charts that begin with "date" use time oriented data, as described above. These charts also support the use of time oriented axis formatting parameters, as described above.

Dataset image parameters will cause your bars to be drawn using tiles of the specified image.

Property	value type	effect
barBaseline	double	bars ascend or descend from this value
barClusterWidth	double	This determines how wide each bar should be. If the value is 1.0, bar 1 will touch bar 2. If the value is 0.5, each bar will take 50% of the available space. If you have more than one data series defined, this value describes the total width of a cluster of bars.
individualColors	true/false	In bar/column charts that normally use only the Dataset color for drawing bars this will determine whether datum colors should be used instead (default is false).
useValueLabels	true/false	Determines whether y values or data labels are used for bar labels.
dataset0y2Values	List	This list of numbers is used to add error bars to each bar (Note: error bars also require "X" values to operate
errorBars	true/false	Determines whether error bars should be displayed

Pie Charts

Pie Charts are drawn with pieApp.



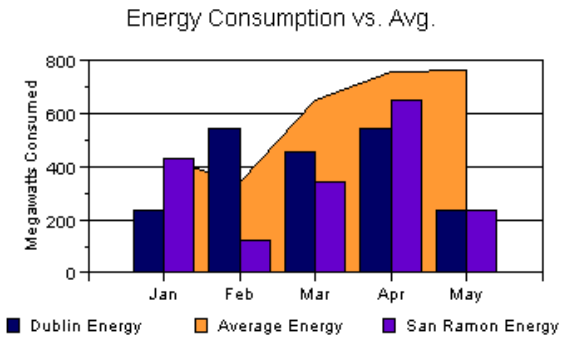
Pie charts can toggle percentage, value, and textual labels. They can also set a beginning angle value, and can set an exploded slice for emphasis. Pie chart colors are defined with the parameter dataset0Colors. Pie charts ignore datasets beyond dataset0.

Pie Chart Properties	value type	effect
explodeSlice	integer	slice number to explode
explodeSlices	list of doubles	This will be list of explosion values for each slice. Explosion values should be between 0 and 1, but generally pretty close to 0. The default value when a slice is exploded with explodeSlice is .05
textLabelsOn	anything	make string labels visible
textLabelsOff	anything	make string labels invisible (default)
valueLabelsOn	anything	make numeric labels visible
valueLabelsOff	anything	make numeric labels invisible (default)
percentLabelsOn	anything	make percentage labels visible (default)
percentLabelsOff	anything	make percentage labels invisible
percentPrecision	integer	the number of digits of precision for Pie percent labels
labelPosition	integer	0: at center of slice, 1: at edge of slice, 2: outside edge of slice with pointer
startDegrees	integer	degrees counterclockwise from 3 o'clock for first slice
xLoc	double	x Location for center of pie (between 0 & 1, default 0.5)
yLoc	double	y Location for center of pie (between 0 & 1, default 0.5)
pieWidth	double	% of window for pie diameter (default .6 = 60%)
pieHeight	double	% of window for pie diameter (default .6 = 60%)
pointerLengths	list	a values to redefine the pointer lengths for external labels. By default, this value is 0.2.
lineColor	Color	redefines the color used for pie slice pointers

Combinations: Bar-Area Chart

BarArea charts layer bars over areas, with shared axes. BarArea charts have variable bar width, and labels that can be toggled on or off. If you don't include a parameter to define X axis labels, this chart will use item labels (param dataset0Labels) beneath each bar. If these labels aren't defined, this chart will display each bar's Y value beneath it. Data series can be assigned to either Bar or Area style charting. Bars draw over areas, and may be stacked or clustered. Areas are always stacked.

barAreaApp

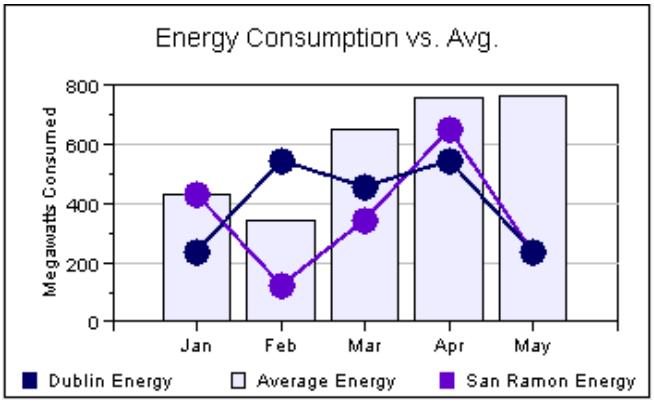


Parameter	value type	effect
datasetNType	Bar|Area	dataset N will be either Bar or Area, based on this value.
stackedBar	true|false	If "true", bars will be stacked, one series upon another.
barBaseline	double	bars ascend or descend from this value
barClusterWidth	double	This determines how wide each bar should be. If the value is 1.0, bar 1 will touch bar 2. If the value is 0.5, each bar will take 50% of the available space. If you have more than one data series defined, this value describes the total width of a cluster of bars.
barLabelsOn	true|false	determines whether labels will be drawn above each bar
barLabelAngle	integer	degrees to rotate bar labels
barLabelPrecision	integer	digits of precision for the bar labels
useValueLabels	true|false	determines whether the bar labels will be dataset labels or y values

Combinations: Bar-Line Chart

Bar-Line charts layer lines over bars, with shared axes. BarLine charts have variable bar width, and labels that can be toggled on or off. If you don't include a parameter to define X axis labels, this chart will use item labels (param dataset0Labels) beneath each bar. If these labels aren't defined, this chart will display each bar's Y value beneath it. Data series can be assigned to either Bar or Line style charting. Lines draw over bars, and bars may be stacked or clustered.

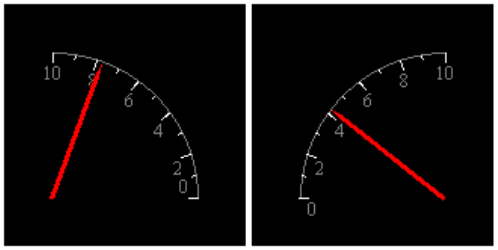
barLineApp

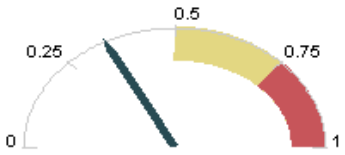


Parameter	value type	effect
datasetNType	Bar|Line	dataset N will be either Bar or Line, based on this value.
stackedBar	true|false	If "true", bars will be stacked, one series upon another.
barBaseline	double	bars ascend or descend from this value
barClusterWidth	double	This determines how wide each bar should be. If the value is 1.0, bar 1 will touch bar 2. If the value is 0.5, each bar will take 50% of the available space. If you have more than one data series defined, this value describes the total width of a cluster of bars.
barLabelsOn	true|false	determines whether labels will be drawn above each bar
barLabelAngle	integer	degrees to rotate bar labels
barLabelPrecision	integer	digits of precision for the bar labels
useValueLabels	true|false	determines whether the bar labels will be dataset labels or y values

Speedos

These include speedoApp and hSpeedoApp. The only difference between these two is that hSpeedoApp adds a history mark in the background; a sort of high water mark.





Speedo charts have adjustable axis locations and styles, as well as adjustable needle styles. This chart can be particularly useful in conjunction with an image background to superimpose a dial and needle on a scanned image of a physical gauge.

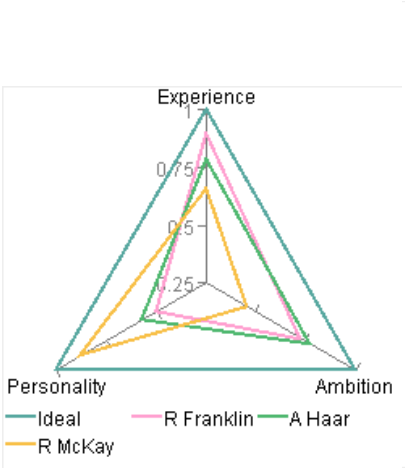
Speedos only use the first value of dataset 0. However, the other values in dataset 0 are considered for building the speedo's scale.

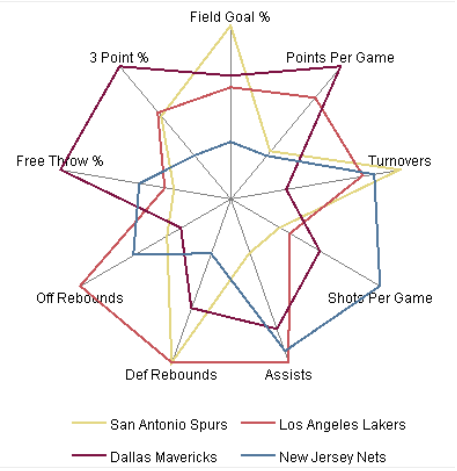
Speedo Chart properties	value type	effect
needleStyle	integer	Kind of needle (default 1) 0 = arrow, 1 = line, 2 = thick arrow, 3 = swept arc
speedoPosition	integer	0 (default) is a mostly complete circle, 1 - 4 are semi circles in various positions, 5-8 are quarter circles in various positions
labelsInside	anything	labels on the inside of the speedo

labelsOutside	anything	labels on the outside of the speedo
watermarkColor	color	for hSpeedoApp, determines the color of the history watermark

Radar Charts

KavaChart "radar charts" or "polar charts" are also called "Kiviat Diagrams". These charts draw multiple spoke axes, with a line for each dataset encircling the center. By default, these charts assume one axis spoke per observation, and they assume that all datasets have the same number of observations.

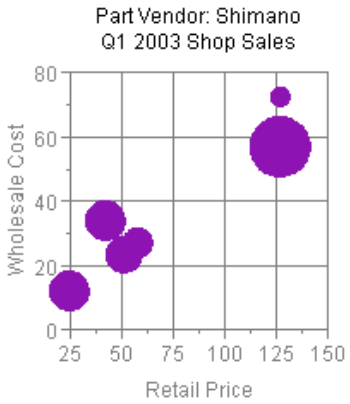




Polar Chart Properties	value type	effect
manualSpoking	true|false	If defined, you are responsible for determining how many "spokes" should be drawn in this chart's axis representation
numSpokes	integer	The number of spokes in this chart's Axis system (default 4)

Bubble Charts

Use bubbleApp to build a bubble chart. This chart draws circles at X,Y values specified by dataset0xValues and dataset0yValues. The size of the circle is determined by dataset0y2Values.



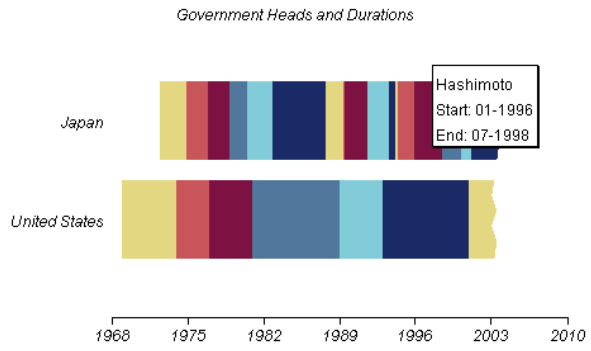
These charts may have filled or hollow circles, crossing X and Y axes, and manual or automatic Z scaling. Z scaling refers to the relative size of the bubbles, based on the overall set of Z (y2) values.

Bubble Chart Properties	value type	effect
zAutoScaleOff	anything	Indicates that you want to set the Z scaling (in terms of a percentage of the Y axis scale.
setZScale	double	Sets the size of bubbles, relative to Y axis units. For example, if the y2 value for a particular bubble is 10 and zScale is set to 2, then the bubble's diameter will be twice as big as a 10 unit increment on the Y axis.
crossAxes	Boolean	Determines whether the X and Y axes should cross. If true, the default crossing value is 0, 0.
xCrossVal	double	Where the Y axis should cross the X axis.
yCrossVal	double	Where the X axis should cross the Y axis.

Gantt Charts

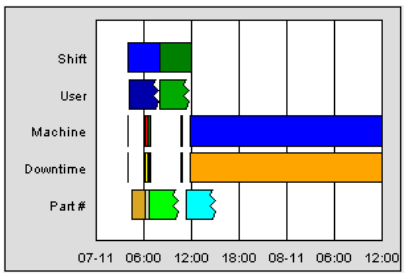
Gantt charts are a specialized chart designed to show when tasks start and end. This sort of chart is particularly useful for resource allocation and project planning, but it can also be used to visually describe the progress of multiple projects or processes.

ganttApp



This chart uses the special params dataset0StartDates and dataset0EndDates to describe the start and end of each colored bar on track "0". Each dataset is arrayed along a single track. In the example above, we're using dataset0 and dataset1 to represent United States and Japanese leader's tenure, respectively. The tooltip label shows the start and end value along with the label (leader's name in this case)

A "discontinuity", or invalid value, like "x" in place of a date creates a torn edge, like the end point on the United States bar, when the property "useTearEdge" is set to "true".



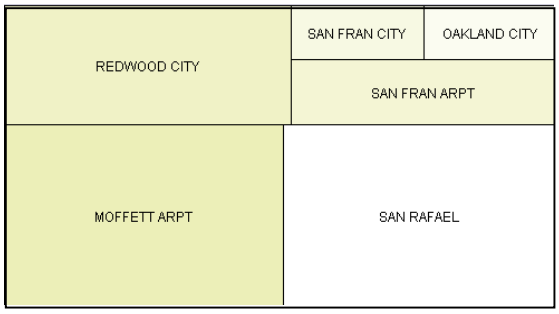
Another special property for this chart, minBarWidth, ensures that very narrow bars, like those in the above, will remain visible.

Parameter	value type	effect
dataset0StartDates	list	A list of dates in “inputDateFormat” format, describing the start times/dates for each item in a particular row. Datasets 0 through 39 are available. Dataset names are used to label the vertical axis.
dataset0EndDates	list	A list of dates in “inputDateFormat” format, describing the ends for each bar segment in a particular row. An un-parseable date, like “XX”, would be interpreted as an incomplete task.
dwellLabelDateFormat	Date format	A format string to describe start and end dates
dwellStartString	String	This string defines the dwell label string for the start date. This string should have 'XX' characters where the date will occur. Default is "Start XX"
dwellEndString	String	This string defines the dwell label string for the end date. Default is "End XX"
dwellIndefiniteString	String	This string defines the dwell label string for an indefinite start/end. Default is "Indefinite"

Sectormap Charts

Sectormap charts are very efficient visuals for displaying certain kinds of data. The size of each square in a sectormap represents its relative size (Y value) within the dataset, and the color of the rectangle represents another factor, such as price change (X value). Each dataset is bounded by a rectangle that represents the Dataset’s overall contribution to Y values for the entire set of datasets.

sectorMapApp



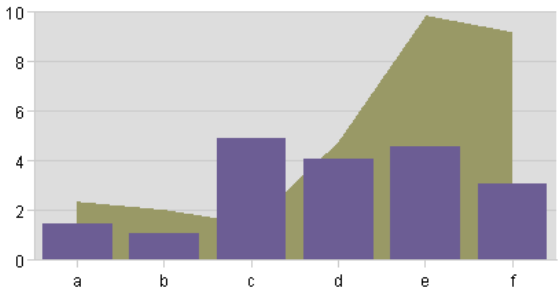
A sectormap could be used to represent financial values in a customer's portfolio, for example, where each data represents a market sector (e.g. finance, transportation, utilities, etc.), and each item in the dataset represents a particular security in that sector. You can tell at a glance how your portfolio is performing, which sectors are doing well in the displayed time period, and which stocks are having the most impact on your portfolio.

Parameter	value type	effect
individualColors	True|false	Determines whether colors should come from "dataset0Colors"
gradientColoring	True|false	Determines whether colors should be auto-graduated from the dataset color to the "secondary color"
sectorSecondaryColor	Color	A second color to be used for gradient coloring
baseColor	Color	A color to be used as a neutral value when "baseValue" is used, giving effectively a 2 dimensional gradient – dataset color to base color to secondary color
baseValue	Double	A value to be used for the baseColor.

Combinations: Bar-Area Chart

BarArea charts layer bars over areas, with shared axes. BarArea charts have variable bar width, and labels that can be toggled on or off. If you don't include a property to define X axis labels, this chart will use item labels (defined with dataset0Labels) beneath each bar. If these labels aren't defined, this chart will display each bar's Y value beneath it. Data series can be assigned to either Bar or Area style charting. Bars draw over areas, and may be stacked or clustered. Areas are always stacked.

barAreaApp

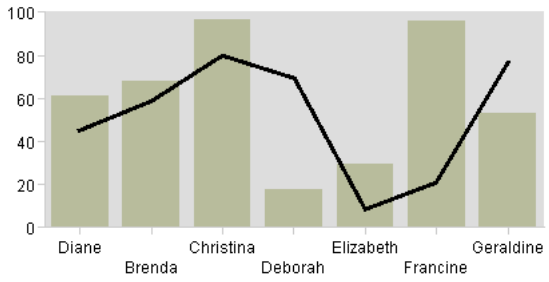


Property	value type	effect
datasetNType	Bar|Area	dataset N will be either Bar or Area, based on this value.
stackedBar	true|false	If "true", bars will be stacked, one series upon another.
barBaseline	double	bars ascend or descend from this value
barClusterWidth	double	This determines how wide each bar should be. If the value is 1.0, bar 1 will touch bar 2. If the value is 0.5, each bar will take 50% of the available space. If you have more than one data series defined, this value describes the total width of a cluster of bars.
barLabelsOn	true|false	determines whether labels will be drawn above each bar
barLabelAngle	integer	degrees to rotate bar labels
barLabelPrecision	integer	digits of precision for the bar labels
useValueLabels	true|false	determines whether the bar labels will be dataset labels or y values

Combinations: Bar-Line Chart

Bar-Line charts layer lines over bars, with shared axes. BarLine charts have variable bar width, and labels that can be toggled on or off. If you don't include a parameter to define X axis labels, this chart will use item labels (param dataset0Labels) beneath each bar. If these labels aren't defined, this chart will display each bar's Y value beneath it. Data series can be assigned to either Bar or Line style charting. Lines draw over bars, and bars may be stacked or clustered.

barLineApp

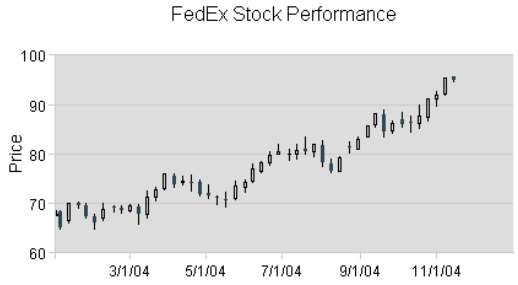


Property	value type	effect
datasetNType	Bar|Line	dataset N will be either Bar or Line, based on this value.
stackedBar	true|false	If "true", bars will be stacked, one series upon another.
barBaseline	double	bars ascend or descend from this value
barClusterWidth	double	This determines how wide each bar should be. If the value is 1.0, bar 1 will touch bar 2. If the value is 0.5, each bar will take 50% of the available space. If you have more than one data series defined, this value describes the total width of a cluster of bars.
barLabelsOn	true|false	determines whether labels will be drawn above each bar
barLabelAngle	integer	degrees to rotate bar labels
barLabelPrecision	integer	digits of precision for the bar labels
useValueLabels	true|false	determines whether the bar labels will be dataset labels or y values

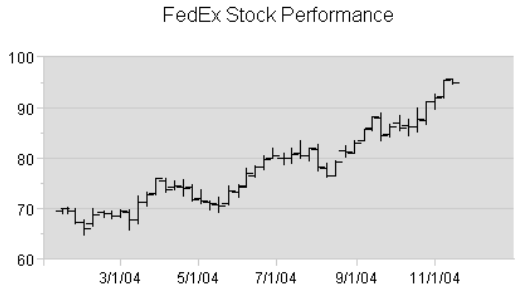
Candlestick and OHLC Charts

This collection includes some standard charts for dealing with financial data: candlestickApp and hLOCApp use 4 Y values for each observation at a single date or time. These are the high, low, open, and close prices for a particular time period.

candlestickApp



hLOCApp



Special properties for these charts:

Parameter	value type	effect

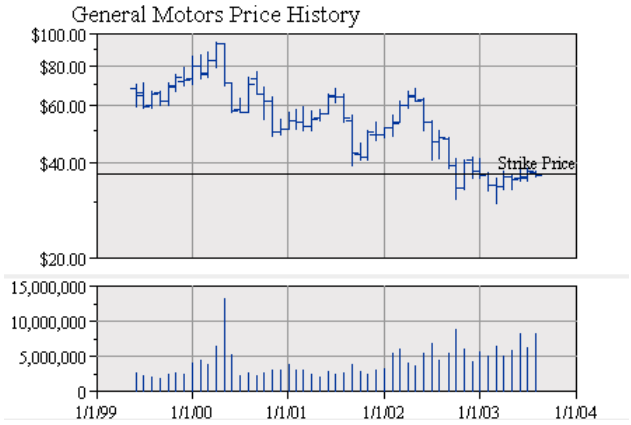
dataset0highValues	list	High price at observed dates
dataset0lowValues	list	Low price at observed dates
dataset0openValues	list	Open price at observed dates
dataset0closeValues	list	Close price at observed dates
dataset0dateValues	list	List of dates in "inputDateFormat"
CustomDatasetHandler	URL	A url containing rows of date,open,high,low.close values

hiLoCloseApp is very similar to Candlestick and OHLC charts, but it uses 3 Y values for each time period. These represent the high, low, and closing prices for a particular time period. Close data is provided with dataset Y values, high data is Y2 and low data is Y3.

Stick Charts

stickApp is similar to a bar chart, but draws a narrow bar, or "stick' at each time period. The width of these bars can be specified in pixels. Multiple datasets do not stack or cluster.

This chart is frequently combined with a hi-lo-close, candlestick or ohlc chart to display price over volume:



Combination Charts

finComboApp combines hiLoClose, line, and stick elements into a single chart with multiple windows. The "splitWindows" parameter determines whether all datasets will appear in a single window, or each dataset should appear in a unique window.

The best way to supply data to these charts is through an implementation of com.ve.kavachart.utility.DataProvider. The datasets provided by this DataProvider should supply datasets that contain com.ve.kavachart..parts.CandlestickDatum classes. See the demos for several examples that supply candlestick data. You can also download one of these DataProviders here:

http://www.kavachart.com/sample_classes/examples.zip

Finance charts can also read data from a URL specified through the parameter "customDatasetHandler". The expected input stream has a column of dates or times in the format specified by the "inputDateFormat" parameter, and then a number of columns of Y data. Each dataset consumes the number of columns appropriate for its data type. For example, in a candlestick chart, each dataset uses the first column as the X axis period, and then uses 4 columns for high, low, open, and close data. A stick would use the first column for the date or time, and then use a single column for each dataset's Y (or price, volume, etc.) values.

Property	value type	effect
datasetNType	HLOC|Stick|Line	dataset N will be either Stick, HLOC, or Line, based on this value. (finComboApp only).
splitWindow	true|false	if true (default) each dataset type will be in a a separate window with an independent Y axis. The X axis will be shared among all dataset types.
stickWidth	Integer	Width (in pixels) of stick bars (stickApp only).

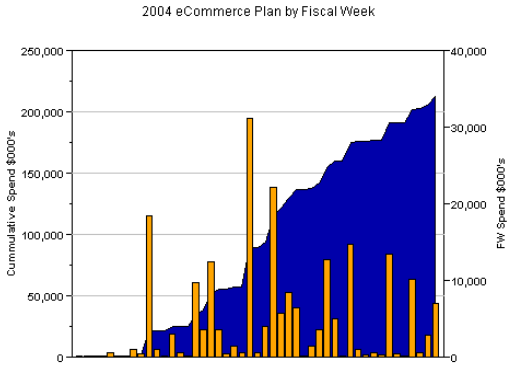
Some of the multiple axis combination charts and time oriented charts are frequently used for financial data.

The KavaChart Enterprise Edition also includes a "kcfinance" package, which is specifically designed to support most common finance charts. This package takes some coding to attach data sources properly, but it's much more sophisticated than the more basic server classes at representing financial data. "Kcfinance" is especially well suited for generating images on a server.

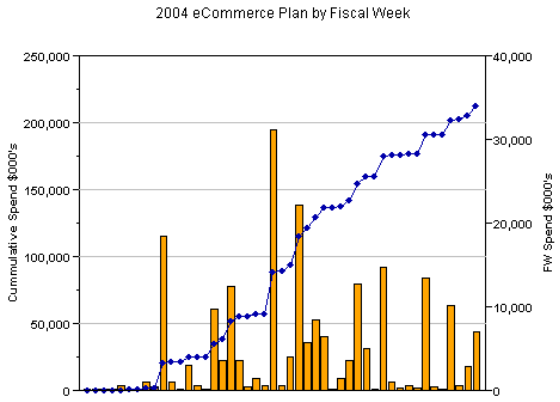
Combinations: Multiple Axis Charts

Many combination charts are more useful if elements are assigned to different Y axes. For example, you might want to compare trends for baseball scores and basketball scores in the same chart. Baseball scores will be much lower, but there still might be some discernable trend. In this case, you could just use twinAxisLineApp to assign baseball scores the the right axis, and basketball scores to the left axis.

twinAxisBarAreaApp: assigns bar data to the left axis and area data to the right (auxAxis).

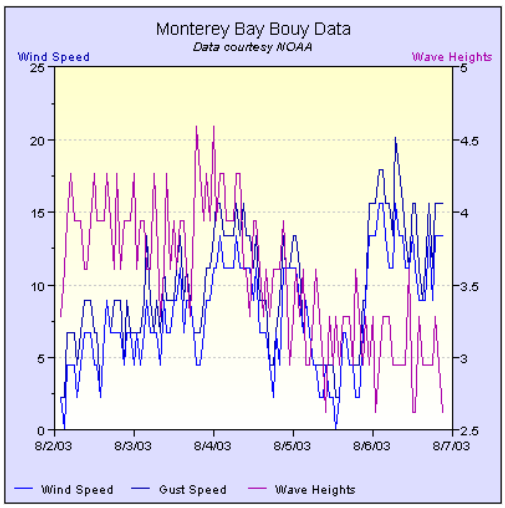


twinAxisBarLineApp: assigns line data to the left axis and bar data to the right (auxAxis).



twinAxisDateComboApp: uses time oriented data, and time oriented axis parameters for the X axis. Datasets can be bar, line, area, or stick, and may be assigned to either left or right axes.

twinAxisDateLineApp: uses time oriented data, and time oriented axis parameters. Datasets are assigned to the left axis by default, and the right (auxAxis) by parameter.



twinAxisLineApp: uses numeric X values. Datasets are assigned to the left axis by default, and the right (auxAxis) by parameter.

twinAxisStackBarLineApp: uses a Line element for the left axis, and a StackBar element for the right axis. Axis assignment is implied by the dataset type.

To change the colors, fonts, title, scaling, etc. for the right axis, use "auxAxis" in place of "yAxis". For example, to set the title, you would use the parameter "auxAxisTitle" for the right, and "yAxisTitle" for the left.

Property	value type	effect
datasetNType	Bar | Line | Area | Stick	This determines the DataRepresentation for datasetN. "Area" is only available for TwinAxisBarAreaApp, Line is not available for TwinAxisBarAreaApp, and so on. Stick is only available for TwinAxisDateComboApp
datasetNonRight	true|false	This determines whether dataset N will be assigned to the standard left axis or the auxilliary right axis. Only applicable to twinAxisDateLineApp, twinAxisLineApp, and twinAxisDateComboApp. Other charts assign one data representation type (e.g. bar) to the primary axis, and the other (e.g. area) to the auxiliary axis.
plotLinesOn	anything	plot lines should display (default). Applicable to all of the Twin Axis Charts except twinAxisBarAreaApp.
plotLinesOff	anything	Create a scatter plot by making plot lines invisible. Appicable to all of the Twin Axis Charts except twinAxisBarAreaApp.
auxPlotLinesOn	anything	plot lines should display (default). Applicable to twinAxisDateLineApp, twinAxisLineApp, and twinAxisDateComboApp.
auxPlotLinesOff	anything	Create a scatter plot by making plot lines invisible. Applicable to twinAxisDateLineApp, twinAxisLineApp, and twinAxisDateComboApp.
barBaseline	double	Bars ascend or descend from this value. Also applicable to Sticks in twinAxisDateComboApp.
barClusterWidth	double	This determines how wide each bar should be. If the value is 1.0

		bar 1 will touch bar 2. If the value is 0.5, each bar will take 50% of the available space. If you have more than one data series defined, this value describes the total width of a cluster of bars.
barLabelsOn	true|false	determines whether labels will be drawn above each bar
barLabelAngle	integer	degrees to rotate bar labels
barLabelPrecision	integer	digits of precision for the bar labels
useValueLabels	true|false	determines whether the bar labels will be dataset labels or y values
areaBaseline	double	sets the baseline value for this area
auxBarBaseline	double	sets the baseline value for the Sticks assigned to the aux axis in TwinAxisDateComboApp
barWidth	integer	sets the width in pixels of the Sticks in TwinAxisDateComboApp
auxBarWidth	integer	sets the width in pixels for the Sticks assigned to the aux axis in TwinAxisDateComboApp

Using a Properties Object or File

In our scriptlet examples so far, we have always used the "setProperty" method to assign property values. There are other ways to set chart properties, and you can combine the various property setting techniques to optimize your server chart implementation.

Property Files

You can store your chart properties in a properties file, which the chart object reads during chart generation. This file is specified with the function "setStyle".

Putting properties in an external file has some significant benefits. If your charts are very different from the default charts, PHP script can become cluttered with calls to "setProperty", making your code appear more complicated than it really is. This sort of inline coding is also not particularly portable, except by doing cut-and-paste operations. Putting all the properties in your code also makes you responsible for the overall style and appearance of the chart, and there might be someone in your organization better suited to creating stylish charts.

You can also use a single properties file for several applications, even though the data acquisition logic may vary dramatically.

A properties file is a text file that looks like this:

```
titleString=Annual Sales
backgroundColor=00ffa6
plotAreaColor=00ff77
dataset0Color=green
yAxisLabelFont=Arial,12,0
```

And so on.

Chapter

7

Image Format Recommendations

KavaChart supports most popular image formats. Which one is best for your application? That depends on the application, but every image encoder has pluses and minuses. These are discussed below

GIF

The GIF89a image format is probably the most widely used on the worldwide web. It provides excellent compression for images that have 256 colors or less, and has some nice features, like transparency and animation.

KavaChart uses a built-in GIF generator when the "imageType" property is set to "gif". This image encoder reduces all images to 256 colors by dithering, and lacks support for transparency.

If you require GIF output with no dithering, or GIF output with transparency support, see the chapter on programming server objects for more information about installing custom image encoders.

JPEG

The JPEG image format is ideal for high color images, such as photographs.

Select an imageType of "j_jpeg" to use a JPEG.

JPEG has some drawbacks when rendering chart output. Because it's designed for photographic images, and uses a lossy compression algorithm, lines (such as axis lines, bar outlines, etc.) may appear blurry. Depending on your situation, however, the image quality might be appropriate.

JPEG also lacks support for transparency. This image format is supported by every web client. Also, this image format requires no special licensing for production use.

PNG

PNG is KavaChart's default image format. Portable Network Graphics (PNG) holds the promise of eventually replacing GIF, and supplanting JPEG for some

applications. If you are generating charts for browsers that support PNG, this is an ideal format for KavaChart output.

Unfortunately, some browsers (notably many older Macintosh browsers) lack support for this format, which may limit its usefulness in your production environment.

Select PNG with the property "imageType=j_png".

Flash

KavaChart can produce Macromedia Flash "movies" that represent charts. Flash has a number of advantages over other image formats: It's a "vector" format, which means that the information is rendered on the client browser. Definitions can be scaled (zoomed in and out by a user), and they carry a built-in set of tooltips and hyperlinks with the flash data stream. KavaChart's highlighting and tooltip behavior is much more animated than the default tooltip labels associated with other image types. Flash output is also "antialiased", which removes the jagged edges associated with the edges of pie charts or diagonal lines. Flash output is very highly compressed, so users perception of performance tends to be very good. The Macromedia Flash plug-in is installed by default on most browsers.

On the other hand, some users are in organizations that don't permit the Flash plug-in. Flash output can't be copy/pasted into other applications. Also some users dislike the slight "blurring" of character edges caused by Flash antialiasing.

In a high-volume production environment, Flash has the advantage of being a vector and polygon format; it doesn't require as much image memory on the server as generating true image formats.

Select the Flash format with "imageType=flash" or "imageType=swf".

SVG

Another vector format available through KavaChart is "Scalable Vector Graphics". SVG support is available in some browser installations, although it's not as widespread as Flash or image support. KavaChart's SVG support is very similar its Flash output: highly animated tooltips and highlighting, highly compressed vector output, etc. You can download and install SVG support from Adobe's SVG web pages.

Although some browsers will recognize and handle SVG as a MIME type, you will generally need to use EMBED tags to handle this data type with Adobe's plug-in.

Select SVG format with "imageType=svg".

BMP

The BMP format is supported in most Windows applications, and is a non-lossy image format, so the images are crisp. Unfortunately, charts created in BMP format are not compressed, and image data can be very large. BMP files are typically not appropriate for web based documents, and are not supported in all browsers.

Select BMP format with “imageType=j_bmp”.

Index