VISUAL ENGINEERING

KavaChart Developer Documentation

KavaChart AlaCarte User Guide

VERSION 5.0

KavaChart AlaCarte User Guide

Table of Contents

Chapter

1

KavaChart Introduction

This chapter provides a broad overview of some of the ways you might use KavaChart to put charts into your application.

What is KavaChart?

KavaChart is a collection of tools for turning numbers into charts. Given one or more series of numbers, KavaChart can create a variety of common charts to help you absorb and interpret the information. KavaChart tools provide robust, well tested components that let software or web site developers translate numbers to graphics with minimal effort.

KavaChart is implemented in pure Java so that it can be used on virtually any computer operating system, ranging from mainframes to PDAs. The charting tools can be used from within HTML pages, Java applications, and various server technologies.

Java programming expertise is not required to use KavaChart, but we assume that our users will have some familiarity with one or more relevant technologies, such as HTML, applets, server scripting, servlets, database access, or object oriented programming. KavaChart is a complement to any of these skills.

Ways to use KavaChart

KavaChart can be used in a variety of ways. Many HTML pages include KavaChart applets that adjust automatically with dynamic data. Other web sites prefer using KavaChart to generate chart images on the server. KavaChart can be used to add charts to Java application programs. KavaChart can also be embedded within other tools, such as EJBs (Enterprise Java Beans) to add charting functions.

KavaChart on the Client: Applets

Although some Java applets are designed to create complete software application functionality, KavaChart applets are designed to behave more like an easy-to-use extension to HTML designed to produce data driven graphics.

KavaChart applets behave like an intelligent image tag that responds to PARAM tags to define the look and data inputs for its graphical output.

Unlike some other charting products, KavaChart applets contain one chart type per applet, and no significant user interface. This means that each applet will download fast and start quick. In some cases, loading a KavaChart applet is faster than downloading an image.

Applets can also contain hyperlinks that will permit you to drill down to more detailed data or to open a reference page. You can have as many unique hyperlinks as you have data points. Hyperlinks can also be used to trigger JavaScript functions on a page.

KavaChart also implements tooltip labels within the applets, so that pointing to a specific bar (or pie slice, or line vertex, or marker, or...) will display the underlying data values and labels.

A chart applet that always shows the same values isn't very interesting, so most KavaChart applet definitions are created with some kind of server scripting language. Since KavaChart's applet data can be defined with simple textual param values, it's a simple matter to create a script that generates these values. When the values change, the chart will change, so building a dynamic chart is as easy as writing a script that generates dynamic page content. And since the values are all simple text, you can use any scripting language; perl, php, servlets, ASP, JSP, your choice.

In addition to params, KavaChart applets can open a URL to retrieve data. They can also poll a URL periodically to see if the data has changed. If the data has changed, the chart will automatically redraw to display the new data.

Enterprise Edition users can also use KavaChart's libraries to build your own custom applets.

The KavaChart Wizard

KavaChart AlaCarte licensees can design charts by hand, using a text editor and documentation about KavaChart’s applet parameters. 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 simulate various kinds of data sources with your chart designs. The Wizard produces complete output templates for a variety of server technologies and data sources.

The KavaChart Wizard is on on-line tool available to all KavaChart users, with enhanced capabilities available to subscribers.

Chapter

2

Quick Start Guide

If you're the kind of person that wants to see results ASAP, follow this quick start guide to make your KavaChart applets productive right away. You can then poke around with the examples and get an overall understanding of KavaChart AlaCarte, and then come back to this guide for a more detailed reference.

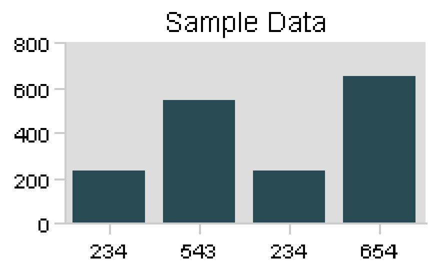
The first step in using KavaChart AlaCarte is to create a web page with an applet definition. Generally it's easier to start with some kind of fixed data in parameters, which you can then turn into live scripted data once you get the applet definiton the way you want it.

Getting Started

An applet definiton consists of a few simple pieces: an APPLET tag, which tells your browser to use Java to interpret the applet information, a CODE attribute, which tells Java which applet you want to use, and ARCHIVE attribute, which tells Java where to find that applet, and WIDTH and HEIGHT attributes, which describe how the applet fits into a page. Here's a simple example:

```
<APPLET CODE=”com.ve.kavachart.applet.barApp”
      ARCHIVE=”applet.barApp.jar”
      WIDTH=”300” HEIGHT=”150”>
</APPLET>
```

This definition produces a simple chart that looks like this:



Let's break down the applet tag, and see what the constituent parts are:

1) **`<APPLET>`** this tells the browser we want to use Java (either the plug-in or Microsoft’s Java for older Windows installations) to interpret the rest of the tag.

2) **`CODE=”com.ve.kavachart.applet.columnApp”`** tells Java that the applet we want to use is named “columnApp”, and that it’s part of KavaChart AlaCarte. All KavaChart applets use the prefix of “com.ve.kavachart.applet”, following Java conventions.

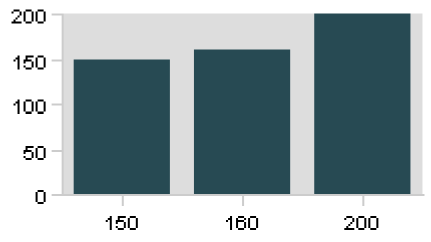
3) **`ARCHIVE=”applet.columnApp.jar”`** tells Java that the applet we’re looking for is contained in the Java Archive file named “applet.columnApp.jar”, which is assumed to be in the same directory as the HTML file, unless you specify the archive as a URL or use a CODEBASE attribute (more about this in the chapters below).

4) **`WIDTH=”300” HEIGHT=”150”`** tells the browser that we want this chart to be 300 x 150 pixels in size. That will be the size of the chart image we specified.

Although this chart is reasonably attractive, it’s also not vey useful. It doesn’t display any meaningful data. To do this, we’d add a PARAM that describes some data. Our new applet definiton:

```
<APPLET CODE=”com.ve.kavachart.applet.barApp”
      ARCHIVE=”applet.barApp.jar”
      WIDTH=”300” HEIGHT=”150”>
      <PARAM NAME=”dataset0yValues” value=”150,160,200”>
</APPLET>
```

And our new chart:



The new chart definition uses actual data from our PARAM tag. We could also use PARAM tags to define multiple datasets, titles, colors, axis scaling, etc., etc. KavaChart supports an extensive range of PARAM definitions that can be edited manually with a text editor, or by using the on-line Chart Wizard.

Creating Dynamic Data

Although there are lots of ways to add data to a KavaChart AlaCarte applet, the easiest method usually involves using some kind of scripting language, like Perl, PHP, VBScript, JSP, Python, etc. to create PARAM values that are tied to your data. For example, our previous chart definition might have looked like this:

```
<APPLET CODE=”com.ve.kavachart.applet.barApp”
  ARCHIVE=”applet.barApp.jar”
  WIDTH=”300” HEIGHT=”150”>
  <PARAM NAME=”dataset0yValues” value=”<?=$my_database_values?>”>
</APPLET>
```

If we use PHP to create a variable named **`$my_database_values`**, then this variable would be processed by PHP while the overall page was created and sent to the browser. Assuming PHP set this variable to “150,160,200”, the output of this applet definition would be identical to our previous example.

Dynamic Data and Chart Intelligence

The beauty of KavaChart AlaCarte, combined with a scripting language, is that KavaChart takes care of managing data display, axis scaling, and chart layout, so that your chart will automatically display an accurate representation of the data you provide. All you really need to do is provide the data and KavaChart does the rest.

Using the Chart Wizard

Although KavaChart provides nice looking output by default for most common chart types, the applets also have many features to help you communicate your data better. You can find out about these features by browsing through the examples or the documentation, or you can use the KavaChart wizard, available at the ve.com web site.

The Wizard generally provides the most productive environment for editing yoru chart definitions. Using the wizard, you can define things like axis scaling behavior, annotation and titling locations, reference lines, etc. using an interactive tool instead of by experimentation. In addition, the wizard will produce output templates that include sample data sources you can modify to suit your own data inputs.

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.

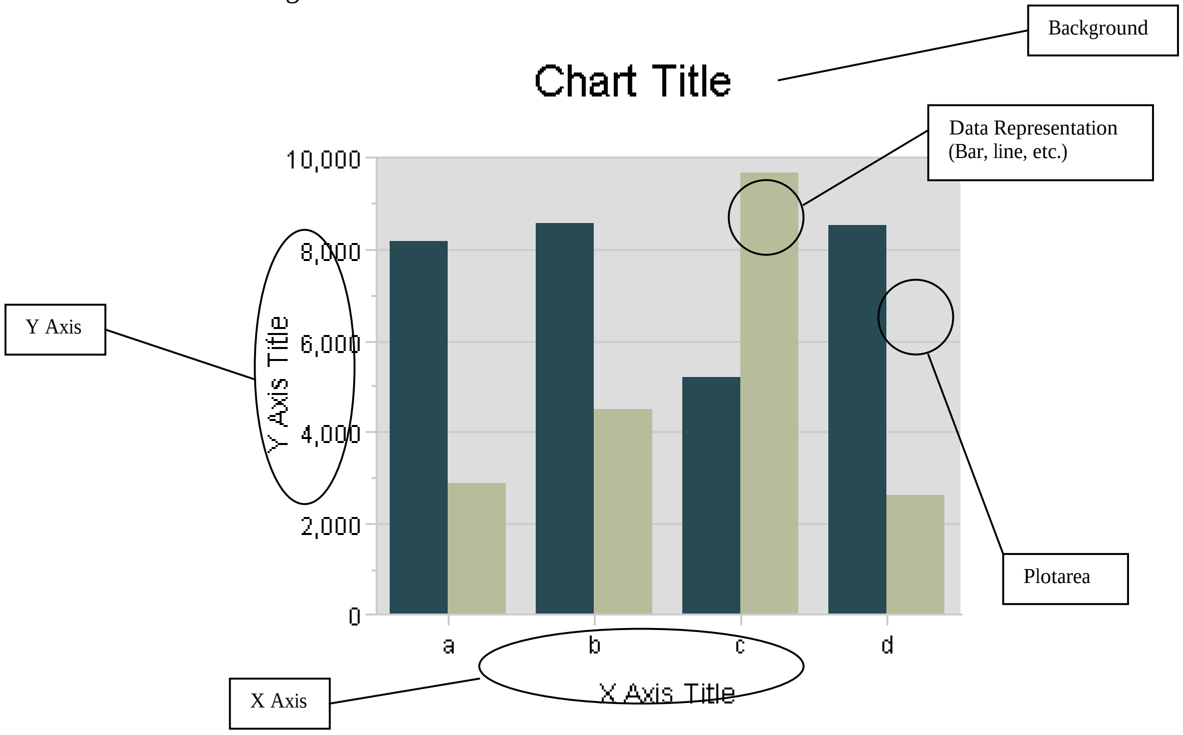
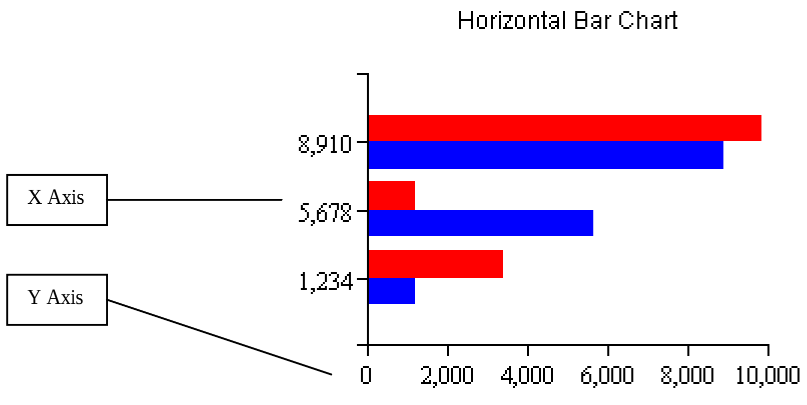
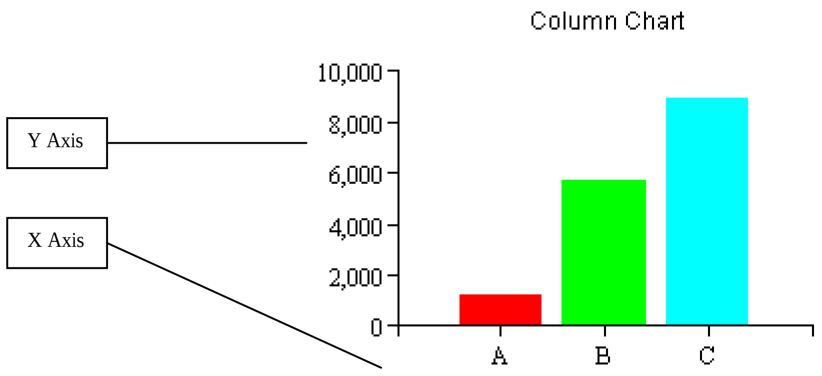


Chart Parts

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.

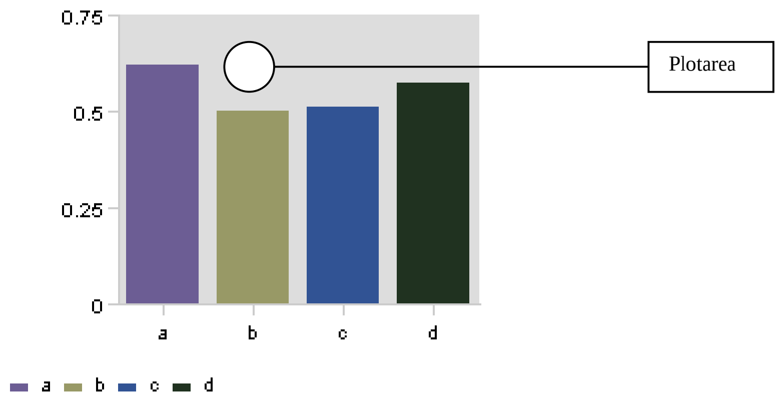
X axes scale for Dataset X values. For some charts, such as a Column chart or a Stacked Column chart, the X axis distributes the data evenly along the X axis, regardless of the Dataset's X values.

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

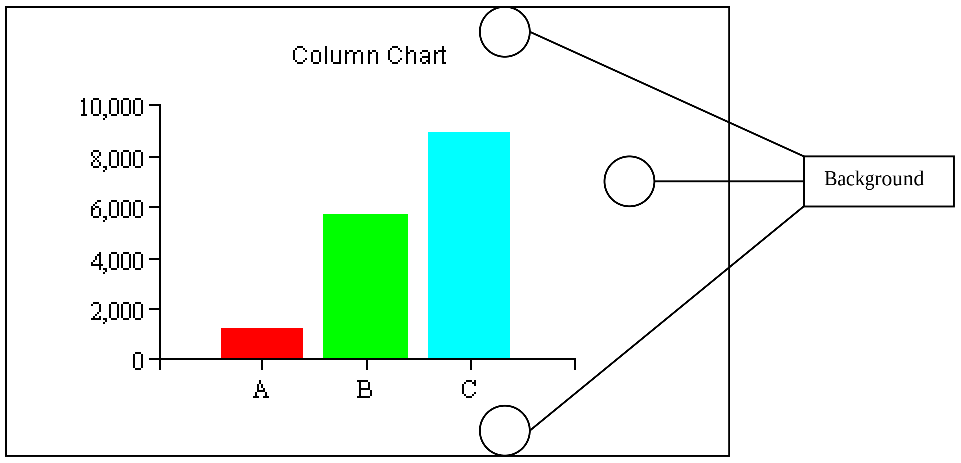


A Plotarea is the region bounded by an X and a Y Axis, which contains a DataRepresentation (such as a Line, Bar, Area, etc.). A Plotarea has a size and location determined by the upper right and lower left corners. The values that define the Plotarea size and location are percentages, relative to the 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.

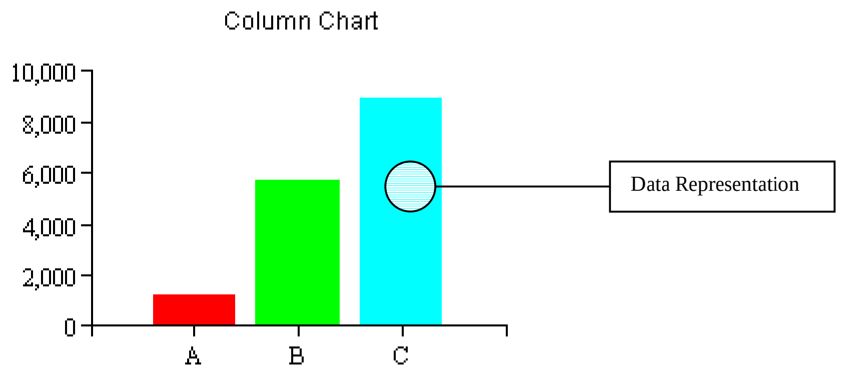
By changing the size and location of your Plotarea you implicitly change the size of your chart's margins. All Axis and DataRepresentation geometries will automatically adjust to accomodate your Plotarea definition.

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

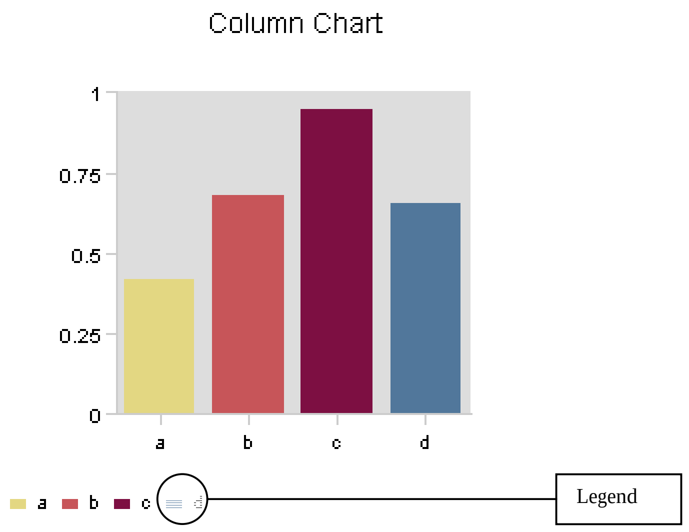


A DataRepresentation is the name KavaChart uses for a variety of objects. These include Line, Area, Bar, and Pie, as well as other more specialized DataRepresentations. These items visually describe a group of Datasets. For example, bar DataRepresentations exist that draw multiple series horizontally 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

4

KavaChart Applet Details

Chart applets provide the quickest way to turn your data into a dynamic chart.

What's an Applet?

Java applets are small programs that typically run within the context of a web browser. When an applet runs, the browser requests the applet's "code" from the server, and then installs and runs that within the browser's Java Virtual Machine, which can be either built-in or a plug-in. The server is responsible only for sending the data that comprises the applet's program code. The browser actually runs the code.

KavaChart organizes charts into a collection of applets, one applet per chart type. This design helps minimize the amount of code that must be downloaded to create a chart image. For example, bar charts don't require any code for supporting pie slices.

Each applet reads from a list of parameters in its HTML definition, and creates a chart image that corresponds to those params. A single HTML page can contain several applets, each one with a unique set of params.

Why Applets?

Applets provide a convenient and automatic way to distribute graphics computing across many machines. While it might not seem like much work to generate a chart image, servicing the requests of many users can create a significant load on your server resources. To distribute this task to page viewers requires only that the server send the applet code and the appropriate params. Since Java support is built into nearly all browsers, this process is completely automatic, and invisible.

Since chart applet appearance and data can be managed with simple text params, it's easy to create server scripts and programs that generate dynamic, data-based charts. All that's required is a server script that generates different params for different charts.

And if your charts include locale-sensitive formatting, such as month names, percentages, or numbers greater than 1,000, KavaChart applets will automatically localize the display for the language and locale settings of the client browser. This can be a powerful tool if you need to support multiple locales from the same data. Of course, you can also specify a locale to make sure your users always see numbers formatted according to a particular convention.

Applets are not completely without drawbacks, however. It's possible for a user to misconfigure their browser's Java support so that applets don't work as expected on a particular machine. It's also possible that your clients include browsers without Java support

Also, the first time a user views an applet, its code must be downloaded from the server. While this process is automatic and doesn't require any special configuration or user confirmation, there may be a slight pause while the Java libraries are loaded and started, and while the actual applet code is downloaded. Once the applet has been viewed, its code is stored in the browser cache, so subsequent views are faster than the initial viewing. KavaChart applets are designed to have a very small code "footprint", to minimize startup time.

Bottom line? Applets are most appropriate for environments that have a known user base; where you understand what browser software will be used to access your information, and are reasonably certain that configuration parameters haven't precluded applet use. Server side charting (KavaChart ProServe) is most appropriate if you need absolute control over the user experience, and have adequate server resources for dynamic imaging.

Anatomy of an applet definition

HTML applet descriptions take a standard form, similar to this:

```
<applet code=MyApplet width=100 height=100>
</applet>
```

When a browser encounters an applet definition like this, it creates a 100 by 100 pixel space within the page, and starts a Java Virtual Machine (JVM) to manage this space. The JVM loads the applet's code from a file called "MyApplet.class" located in the same document base as the HTML containing this applet. If MyApplet.class needs additional resources, those resources are loaded automatically by the JVM.

KavaChart applet definitions are only slightly more complicated. Here's a simple example:

```
<applet code=com.ve.kavachart.applet.barApp  width=300 height=200>
       <param name=dataset0yValues value="234,432,234">
```

```
</applet>
```

This example has two important differences. First, the JVM isn't looking for a class file named "com.ve.kavachart.applet.barApp.class", even though it looks that way. Instead, this file is located in "com\ve\kavachart\applet\barApp.class", using Java's namespace conventions. Internally, this file identifies itself as part of a Java "package" called com.ve.kavachart.applet, and the files in that package must all be located in the directory or folder named "com\ve\kavachart\applet".

Second, this example uses a "param" statement to define some data. KavaChart applets understand an extensive list of parameters for defining just about every aspect of a chart, including colors, layout, fonts, titles, axis management, and so on.

It's also important to note that barApp doesn't include everything needed to draw bar charts. It uses a set of other class files located in com\ve\kavachart\applet and com\ve\kavachart\chart. These include com.ve.kavachart.applet.ChartAppShell, which manages applet drawing, printing, etc., com.ve.kavachart.utility.ParameterParser, which reads and interprets applet parameters, com.ve.kavachart.chart.BarChart, which contains bar chart drawing logic, com.ve.kavachart.chart.Axis, which draws axes, and so on. The JVM will automatically figure out which support files are needed, and will pull them from the subdirectories.

In practice, however, you'll be using KavaChart's pre-built archive files, like this:

```
<applet code=com.ve.kavachart.applet.barApp archive=applet.barApp.jar
    width=300 height=200>
     <param name=dataset0yValues value="234,432,234">
</applet>
```

In this example, the browser looks file a file named "barApp.jar" for the applet's code. This file is a ZIP archive that contains all the class files required by com.ve.kavachart.applet.barApp. If you examine this archive with a tool like "jar" or "WinZip", you'll see that it contains part of the com.ve.kavachart directory structure, including com\ve\kavachart\applet\barApp.class.

You'll find a JAR file for each applet in the KavaChart AlaCarte collection you downloaded. The JAR files are named "applet.className", where "className" is the name of the specific Java applet you're using.

Using CODEBASE

Another important applet definition attribute is **CODEBASE**. Here's an example:

```
<applet code=com.ve.kavachart.applet.barApp
       archive=barApp.jar codebase="/javastuff/jars/"
       width=300 height=200>
<param name=dataset0yValues value="234,432,234">
</applet>
```

CODEBASE causes the browser to look in a specific location for its code resources. In this example, the browser will look for /javastuff/jars/barApp.jar to find com.ve.kavachart.applet.barApp. This is particularly useful when your page is generated from a script or a servlet, so that the document base is ambiguous. Since KavaChart applets are generally used in conjunction with a script or a servlet to define the data parameters dynamically, the CODEBASE attribute is especially useful for KavaChart applets.

Another side effect of using codebase might not be so obvious. Java applets are designed to be extremely secure. This is accomplished by limiting what an applet can do. One of the limitations restricts applet access to any server resources outside the CODEBASE. This means that a KavaChart applet param that specifies an image texture or marker outside the CODEBASE will fail. The same restriction will affect URL data params, background images, and fill textures. The solution is to set up your codebase in a way that permits access to the server resources you need. Here's how we can change the previous definition to give us more access to server resources:

```
<applet code=com.ve.kavachart.applet.barApp
archive="javastuff/jars/barApp.jar" codebase="/" width=300
height=200>
<param name=dataset0yValues value="234,432,234">
<param name=dataset0Image value="images/marble.jpg">
</applet>
```

By moving codebase to the server's root directory, we get access to the entire server. And, by expanding our archive definition, we can still point to the same server directory and archive file.

KavaChart Applets and Data

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.

How do you get all this information into your applet? Generally you'll do this using applet dataset params. Here's an example of an applet with a simple data definition:

```
<applet code=com.ve.kavachart.applet.columnApp
  archive=”applet.columnApp” width=300 height=200>
    <param name=dataset0yValues value="234,321,234">
</applet>
```

This applet uses a single parameter to define the "Y" values for dataset 0. To add another series, we'd just add another param, using dataset1yValues. To add another bar to our chart, we'd just add another number to the list "234,321,234".

In this applet we don't need X values, Y2 values or anything else, because we're just dealing with a simple bar chart. If we wanted to add some labels, we could do this:

```
<applet code=com.ve.kavachart.applet.columnApp
   archive=”applet.columnApp” width=300 height=200>
    <param name=dataset0yValues value="234,321,234">
    <param name=dataset0Labels value="a,b,c">
</applet>
```

It doesn't matter what order the params are in.

The item labels can be used in various ways. For example, right now, the chart will use these labels along the horizontal axis to label each bar. However if we add the param "labelsOn" and set the value to "true", we'll get labels at the top of each bar. The same applet definition can be used for pie charts, line charts, or any other kind of chart by changing the "code=" portion of our applet definition.

Params are available for dataset0yValues, dataset0xValues, dataset0y2Values, dataset0y3Values, and dataset0Labels for datasets 0 through 39. This is a convenient way to generate dynamic charts if you're building your HTML output with a script or a servlet. Just make the param values dynamic, and the charts will reflect up to the minute data. For example, if you are using a JSP, do something like this:

JSP SAMPLE

```
<%
      String getSomeNumbersHere(){
            return "123,432,123";
      }
      String getSomeLabelsHere(){
            return"a,b,c";
      }
%>
<applet code=com.ve.kavachart.applet.columnApp width=300
height=200>
<param name=dataset0yValues
value=<%=getSomeNumbersHere()%>>
```

```
<param name=dataset0Labels value=<%=getSomeLabelsHere()%>>
</applet>
```

Of course, your JSP will do something much more useful, like display actual data, but the basic concept is the same.

Dataset Parameters

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.

Parameter 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)

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 param dataset0dateValues, like this:

```
<applet code=com.ve.kavachart.applet.dateLineApp width=300
   height=200>
      <param name=dataset0yValues value="234,321,234">
      <param name=dataset0dateValues
   value="01/01/02,02/01/02,03/01/02">
</applet>
```

This param translates the dates into a form usable by Java classes 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:

```
<applet code=com.ve.kavachart.applet.dateLineApp width=300
height=200>
<param name=dataset0yValues value="234,321,234">
<param name=dataset0dateValues
```

```
value="01/01/02,02/01/02,03/01/02">
<param name=inputDateFormat value="MM/dd/yy">
</applet>
```

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

Tip:

If you're generating dynamic applet data from a JSP that uses JDBC, you can probably use the param dataset0xValues. Assuming your observation dates are java.sql.Date classes, just use the "getTime()" method to pass the raw numeric information into the applet instead of formatting the output to match an applet input format.

Time and date oriented charts have special parameters for managing axes, which are listed later in this chapter.

URL Datasets

KavaChart applets can also read data from a URL. In this case, the applet runs without data in the "param" tags, but retrieves numbers and labels from the server. This is particularly useful if you expect your users to keep a page open for a while, since you can instruct the applet to check for new data periodically, and it will automatically update when the data changes.

Here's a sample:

```
<applet code=com.ve.kavachart.applet.columnApp width=300
height=200>
<param name=dataset0yURL value="http://yourserver/cgi-
bin/NumberGenerator.cgi">
</applet>
```

This applet expects the URL "http://yourserver/cgi-bin/NumberGenerator.cgi" to return a delimited list of numbers, like this:

```
123,432,123
```

The default delimiter is comma, but you can also use the param named "delimiter" to change this to something else.

You can also point to a file, a servlet, a JSP, or any other URL that returns a delimited list of numbers.

This kind of data retrieval becomes much more interesting if you add another param, "networkInterval". This param instructs the applet to re-read the URL after the specified period for a new data list. The new data replaces the entire dataset with new data, rather than appending data. To make your applet re-read data every 5 seconds, do this:

```
<applet code=com.ve.kavachart.applet.columnApp width=300
height=200>
<param name=dataset0yURL value="http://yourserver/cgi-
bin/NumberGenerator.cgi">
<param name=networkInterval value="5">
</applet>
```

Other params exist for X values (dataset0xURL), Y2 values (dataset0y2URL0, Y3 values (dataset0y3URL0), data labels (dataset0URLLabels), and for combined X and Y data, in the form of x1,y1,x2,y2,... (dataset0xyURL). Of course, if you have a large number of datasets, the applet might be calling many scripts to get all this information, so KavaChart applets also support reading blocks of data in various forms.

These block params include the simple param URLDataBlock, which points to lists of Y values, like this:

```
123,432,123
432,123,452
123,432,123
```

Where each line is a new dataset, and each item in the line is another Y value.

You can also use a list of row oriented data with URLDataRows, which is similar to URLDataBlock, but interprets each line as containing both X and Y values. If you have column-oriented data, you can use URLDataColumns. For efficiency, this format expects to have a single number on the first line that specifies the number of rows in the output. The first two columns are dataset 0's X and Y values. The next two columns are dataset 1's X and Y values, and so on. The block data readers expect all datasets to have the same number of observations.

If you're using a date oriented chart, there's a special param called "customDatasetHandler". This param points to a URL that returns data in a format like this:

```
12/25/02, 3, 2, 5
12/26/02, 4, 6, 3
12/27/02, 7, 5, 8
...
```

Each line in this stream consists of a timestamp (a date in our example), together with a Y value for each dataset you're using. Our example has 3 datasets.

URL Dataset Parameters

The table below describes KavaChart's built-in URL dataset parameters.

Parameter Name	Type	Effect
dataset0xURL	URL	URL for a file of comma separated X values for dataset 0
dataset0yURL	URL	URL for a file of comma separated list of Y values for dataset 0
dataset0URLLabels	list	comma separated labels for URLs
URLDataBlock	URL	URL for a file of comma separated Y values. Each line in this file is assumed to be a unique dataset.
URLXYDataRows	URL	URL for a file of comma separated XY pairs arranged into rows. Each row represents a single dataset, with values arranged as x1, y1, x2, y2, x3, y3...
URLXYDataColumns	URL	URL for a file of comma separated XY pairs arranged into columns. This file must contain a single value as the first line of the file, specifying the number of observations (rows) in the file. The first two columns are dataset 1, the second two columns are dataset 2, etc.
networkInterval	integer	Number of seconds to wait before re-reading URL datasets
customDatasetHandler	string	information passed to a user-defined dataset handling routine "void getMyDatasets(String str)" Note: in the case of time oriented charts, this is the URL for a data file that contains columnar Date information

internalData	True|false	If this parameter is set to "true", the user-defined method "getDataset" will be called repeatedly until a null value is returned.

Special time oriented URL parameters

Time and date applets use some additional special parameters to limit views on incoming data, add incremental data, and so on. These are in addition to the "inputDateFormat" parameter, which should always be used to help the applet parse incoming data.

Parameter	Type	Effect
startData	String	Ignore any data earlier than this time/date. Note: this string is passed into Java's "Date" class to be translated into a machine independent time stamp. Many time-stamp formats will work. If you need to use a specific input format, see the Date Format section below.
endData	String	Ignore any data later than this time/date. Note: this string is passed into Java's "Date" class to be translated into a machine independent time stamp. Many time-stamp formats will work. If you need to use a specific input format, see the "Incoming Date Formats" section above.
inputDateFormat	String	Use this pattern to read any incoming data. For more information on how to construct this string, see the "Incoming Date Formats" section above.
incrementalDataURL	URL	A URL to poll for additional data points. Date oriented charts read initial data from the URL or file specified in "customDatasetHandler". Subsequent observations can be added to applets by polling the URL specified in this parameter. This URL will be checked each networkInterval.

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:

```
<applet code=com.ve.kavachart.applet.disLineApp
                            width=300 height=200>
<param name=dataset0yValues
value="23,32,45,43,23,65,45,x,54,34,76,45,78,54,23">
</applet>
```

CODEBASE, again

There are some important restrictions placed on Java applets that you'll need to be aware of if you're using URL data sources. Java applets are designed to be very secure, and run in a so-called "sandbox" that prevents access to certain kinds of resources that are available to other programs. In particular, applets are not permitted to open URLs outside their "CODEBASE". This is done for a very good reason, to prevent applets from accessing arbitrary network locations, but it can also lead to failed URL data feeds. If your URL data doesn't come from a source "beneath" the applet's document base somewhere, see the discussion above on CODEBASE to see how to work around this issue.

Color and Style Parameters

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

Color and style parameters take different kinds of values. The table below gives you some examples of what these values should be.

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.	`<param name=”legendTexture” value=”1”>`
Double	A real number value, like 0.25. Generally, these values are expressed in terms of a percentage of the overall chart size.	`<param name=”plotAreaBottom” value=”0.12”>`
Font	font parameters include information for the font name, the font size, and the font style. Any valid Java font works here, but we start with a default of TimesRoman 12pt in most cases to ensure that the font is available. 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.	`<param name=”xAxisLabelFont” value=”Arial,18,2”>`
Color	this field expects a color name, or a hexadecimal color definition (in RGB). Valid colors names in these applets 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.	`<param name=”titleColor” value=”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.	`<param name=dataset0Colors value=”green,red,ff00aa”>`
String	A text string	`<param name=titleString value=”hello, world”>`
url	These fields expect some URL specification within your applet’s CODEBASE. Relative or absolute URLs are OK.	`<param name=”backgroundImage” value=http://me/coolpic.jpg>`

Boolean	Either “true” or “false”	**<param name=”outlineLegend” value=”false”>**
Anything	Some parameters can take any value. The applet just wants to know if the parameter has been defined	**<param name=”3D” value=”yeah, sure”>**

General Color and Font Parameters

The first set of parameters applies to all charts. These parameters 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”
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

labelFormat	integer	deprecated - by default, servlets and applets will use the locale to determine how numbers should be formatted. You can override this with labelFormat for compatibility with older releases of KavaChart. Also see: locale
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
legendHorizontal	anything	legend icons in horizontal list (default)
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 legend's 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 hatching, and -1 to use the legend image 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 (default 0.8)
plotAreaBottom	double	bottom of the plotting area (default 0.2)
plotAreaRight	double	right side of the plotting area (default 0.8)
plotAreaLeft	double	left side of the plotting area (default 0.2)
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). Use this property to define line markers for scatter plots.
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)
locale	String	KavaChart automatically localizes your charts for the locale of the Java Virtual Machine that creates the chart. For applets, this means your charts will automatically change things like month labels, number formatting, and so on, depending on whether your viewers are in, say, France, or Japan. You can override the locale with this parameter if you create a signed applet. Generally, locale changes are disallowed by the applet SecurityManager. Valid locales include canada, canada_french, china, chinese, english, france, french, german, germany, italian, italy, japan, japanese, korea, korean, prc, simplified_chinese, taiwan, traditional_chinese, uk, and us. You can also create a locale using two letter language codes and country codes in this format: langageCode_countryCode (for example "en_US" denotes english/U.S.).
delimiter	String	the separator character for list parameters. Default is comma (e.g. "123.432.123").
defaultFont	Font	A new default font for your charts. This parameter overrides the default font setting for KavaChart graphs. This parameter sets a new default for all KavaChart graphics running within the Java Virtual Machine in the current session, so you should use it cautiously. Its primary value is for settings that wish to start with consistent font usage for all charts.
backgroundImage	URL or file name	Charts can replace the solid background color with a GIF or JPEG image for added effect.

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 printed with the version number as the chart 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 Parameters

The following tables contain parameters for adjusting axes. Line, area, bar, and their derivatives use these parameters. Axis parameters consist of a set of parameters and an option list. The option list is followed by a separate table that describes detailed axis options.

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: *param name=xAxisOptions value="gridOff, tickOff, lineOn"*. If you're modifying an auxiliary Y axis (such as in a chart that has left and right axes), use auxAxisOptions.

yAxisOptions (xAxisOptions)	Effect
autoScale	automatically create axis scale (default)
noAutoScale	axis scale defined in applet parameters
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 will use a localized percentage representation for the axis labels (not valid for time and label axes)
currencyLabels	This axis will use a localized currency representation for the axis labels (not valid for time and label axes)

Detailed Axis Parameters

If you're modifying an X Axis (usually on the top or bottom of a chart), use xAxis*ParameterName* instead of yAxis*ParameterName*. 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*ParameterName* instead of yAxis*ParameterName*.

Axis Parameter	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, although you can change this by making a minor modification to the applets.
yAxisLabelAngle	integer	label rotation in degrees (default 0). Note: rotations of 0 and

		90 degrees will be the most readable
yAxisLabelFormat	0:default, 1:Comma, 2:European	(default 0) Note: by default, charts will automatically localize formats based on the Java Virtual Machine running the chart. (deprecated – see “locale”)
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. This will cause the axis to behave as if all your data starts at your specification, but ends wherever the real data ends.

Date and Time Axis Parameters

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 Parameters	Type	Effect
startDate	string	time/date for axis starting value. Note: this string is passed into Java's "Date" class to be translated into a machine independent time stamp. Many time-stamp formats will work. If you need to use a specific input format, see the Date Format section above.
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. Your applets will automatically use, for example, Japanese month names for browsers in Japan, and German month names for browsers in Germany. Similarly, applets might choose a yy/mm label for one locale, and mm/yy for another locale. This parameter lets you override the axis labels to use your specific formatting instructions. See the Date Format section 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 applet and servlets use the timezone of their jvm. This may be incorrect in some cases, for example if my servlet 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 applet and servlets use the timezone of their jvm. This may be incorrect in some cases, for example if my data is based in New York, my client's applet is parsing time data in California, and I want my user to see the data in real-time, then this parameter can be used to change the way the data is parsed. This is an alternate to inputting the timezone in your date strings. 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").

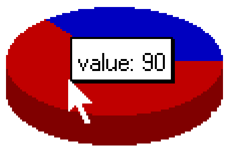
Dataset Related Color and Style Parameters

Dataset colors and styles are very important to KavaChart applets. 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.
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		image to use for this dataset's markers (default none). Use this property to define line markers for scatter plots.
dataset0Images	list of URLs	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)

All KavaChart applets share some additional parameters that control what information appears in the chart’s dwell labels. These labels are similar to tooltips, and appear when the cursor is placed over a data item.



You can control what appears in this label: X values, Y values, item labels, dataset labels, and so on. Here are the parameters that format your dwell labels:

Parameter	value type	effect
dwellLabelsOn	true/false	Tells the applet whether to use dwell labels.
dwellUseLabelString	true/false	Tells the applet whether to use each datapoint's label as a part of the popup dwell labels.
dwellUseXValue	true/false	Tells the applet whether to use each datapoint's X value as a part of the popup dwell labels.
dwellUseYValue	true/false	Tells the applet whether to use each datapoint's Y value as a part of the popup dwell labels.
dwellUseDatasetName	true/false	Tells the applet whether to use dataset names in the popup dwell labels.
dwellXString	String	A text string containing the character "#" to add descriptive text to the dwell label X value. Example: "Category #"
dwellYString	String	A text string containing the character "#" to add descriptive text to the dwell label Y value. Example: "Unit Sales: $#"
dwellLabelDateFormat	String	A format string for describing dates in the dwell label (e.g. yyyy = 2001). This parameter is used only by time oriented charts, and uses the formatting strings described above.
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 or 123,45.

Hyperlinks

Hyperlinks permit you to click on portions of your chart and open URLs. This can be used as a drill-down mechanism to obtain more detailed information about a particular item, or it can simply open any URL.

By default, hyperlinks open in a new browser frame. You can specify a target frame name, however, or you can specify a target of "_this" to open your URLs in the current frame.

Note:

By preceding your hyperlink with "javascript:" you can launch a JavaScript method on the same page. For example: "javascript:alert('hey')" would launch an alert box.

Because hyperlinks are tied to individual data items, you must provide a list of URLs for each dataset, similar to the way you specify data labels.

Parameter	value type	effect
dataset0Links	list	a list of URLs for hyperlinks from dataset0. Datasets 0 through 39 available.
target	String	Target frame for hyperlink drill-down results

Chapter

5

The Applets and Their Parameters

KavaChart AlaCarte includes 4 applet collections. This chapter details the specific parameters available to each chart in each specific collection.

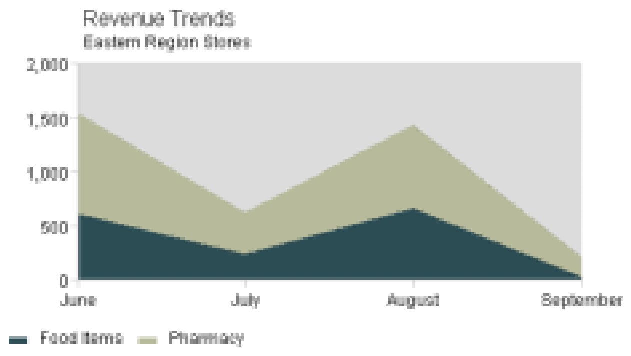
Each applet has a few parameters that deal only with that chart type. For example, pie charts have a parameter that lets you set the starting angle of the pie. This parameter doesn't make sense for bar charts.

Basic Applet Collection

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.

com.ve.kavachart.applet.areaApp

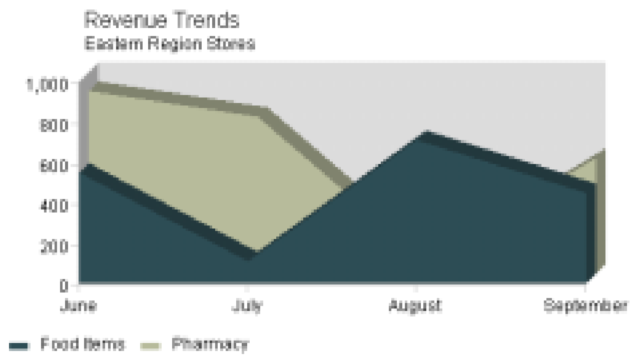


AreaApp ignores your X value specifications and assumes the values are 0, 1, 2, 3, … This ensures that the areas will align properly. Use the xAxisLabels parameter to specify your actual labels.

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:

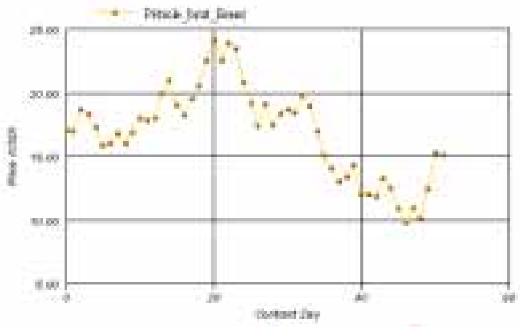


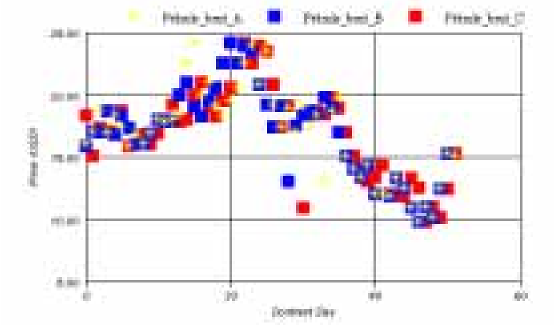
Parameter	value type	effect
baseline	double	sets the baseline value for this area
stackAreas	true/false	determines whether the areas will be stacked on top of each other (default is true)

Line and Scatter Charts

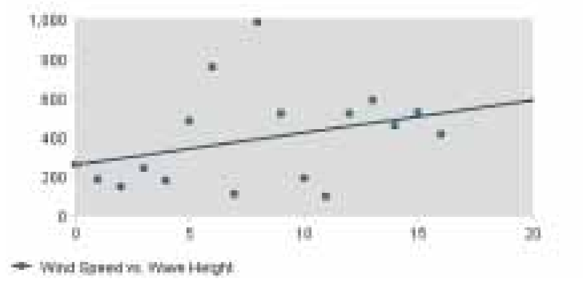
These charts include:

com.ve.kavachart.applet.lineApp

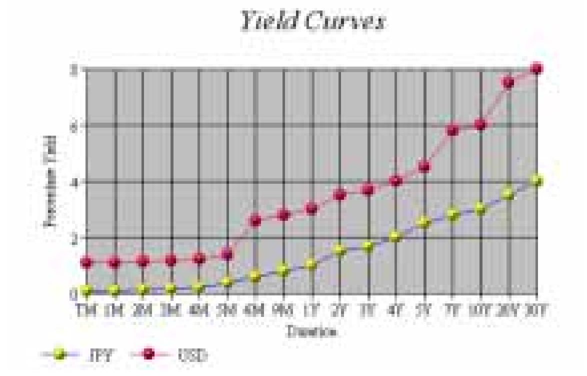




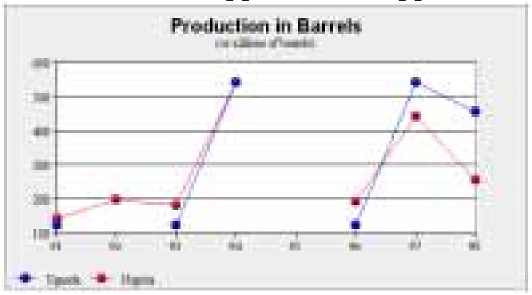
com.ve.kavachart.applet.regressApp



com.ve.kavachart.applet.labelLineApp



com.ve.kavachart.applet.disLineApp



com.ve.kavachart.applet.disLabelLineApp

com.ve.kavachart.applet.zoomLine

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

Applets that begin with “dis”, such as disLineApp and disLabelLineApp support discontinuous data. They will create line breaks where data is missing. See the “data” section in the previous chapter to understand how to define discontinuities

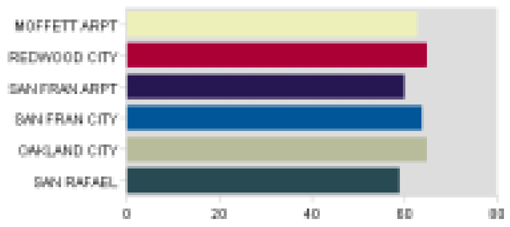
RegressApp performs a simple linear regression calculation using the data values passed into the applet. Markers appear at the actual data points, while the line is drawn according to the regression’s prediction. This is a classic “scatter plot” that shows positive, negative, or no correlation, and gives visual feedback about the strength of that correlation.

Parameter	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). (dataset0Markers, dataset0Colors, etc.)

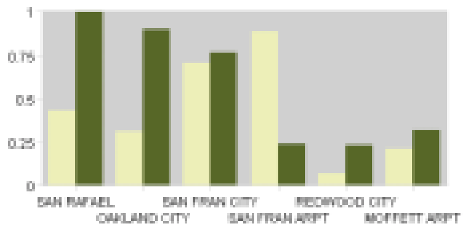
Bar and Column Charts

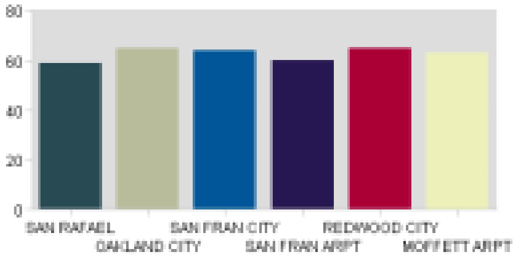
This category includes both charts with vertical and horizontal bars, as well as hi-lo bars. The applets are:

com.ve.kavachart.applet.barApp

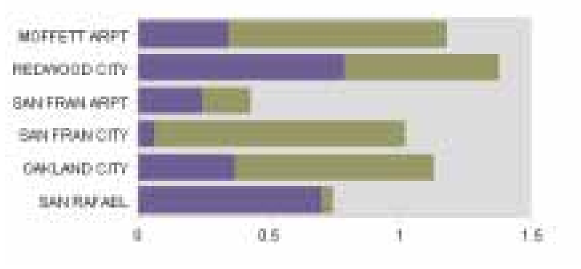


com.ve.kavachart.applet.columnApp

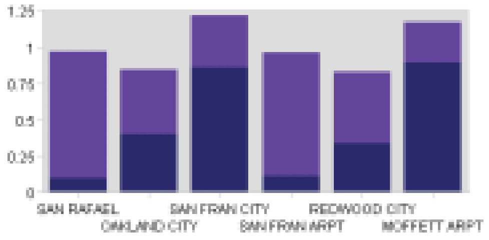




com.ve.kavachart.applet.stackBarApp



com.ve.kavachart.applet.stackColumnApp



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 parameter "individualColors" to true, and define the colors with "dataset0Colors".

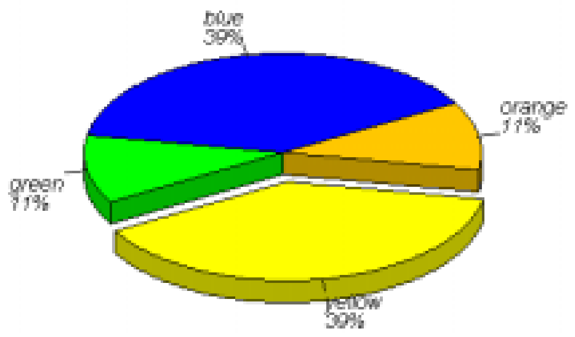
StackBarApp and stackColumnApp stack datasets instead of clustering them. This is useful to display a cumulative summary along with the individual data.

If you're in a Java 2 or better environment, dataset image parameters will cause your bars to be drawn using tiles of the specified image.

Parameter	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 the bar labels are the dataset labels or just the y values (default is false)
dataset0y2Values	List	This list of numbers is used to add error bars to each bar. (note: you must also set X values e.g. 0,1,2,3… for error bars to appear)
errorBars	True/false	Determines whether error bars should appear

Pie Charts

Pie Charts include com.ve.kavachart.applet.pieApp, and com.ve.kavachart.applet.spinningPie. Spinning pie is a variation that lets users grab a slice and rotate the pie around.



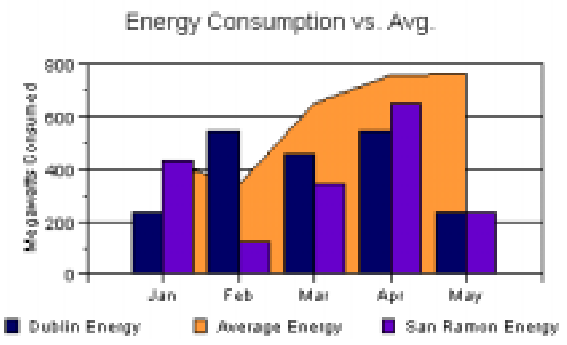
Pie charts can toggle percentage, value, and textual labels. They can also set a beginning angle value, and can set an exploded slice for emphasis. Pie chart colors are defined with the parameter dataset0Colors. Pie charts ignore datasets beyond dataset0.

Pie Chart Parameters	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 list of 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.

com.ve.kavachart.applet.barAreaApp

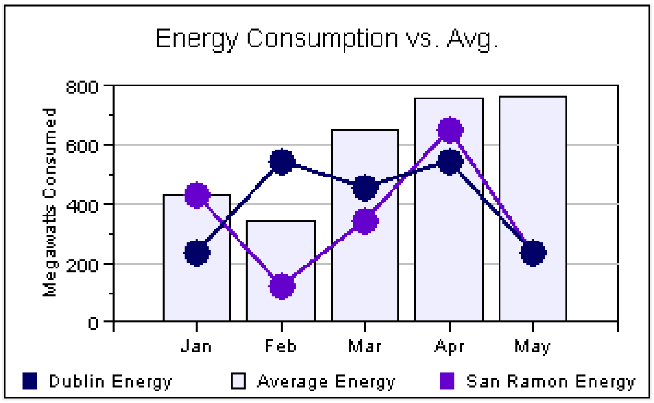


Parameter	value type	effect
datasetNType	Bar|Area	dataset N will be either Bar or Area, based on this value.
stackedBar	true|false	If "true", bars will be stacked, one series upon another.
barBaseline	double	bars ascend or descend from this value
barClusterWidth	double	This determines how wide each bar 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.

com.ve.kavachart.applet.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

Interactive Applets: spinningPie

SpinningPie (com.ve.kavachart.applet.spinningPie) is a version of pieApp that lets users grab the edges of a pie and rotate it . This may have some practical purposes, but it’s primarily use just stresses that this applet is live, and not a preconsructed image.

Interactive Applets: zoomLine

ZoomLine (com.ve.kavachart.applet.zoomLine) extends the basic functionality of lineApp to support data zooming. By dragging the mouse over a region of the chart, users create a rubberband effect, which shows the approximate window that will be displayed when the mouse button is released.

A right click, or alt-click on some systems, will reset the zoom level to no zooming.

Interactive Applets: scrollingLine

This applet (com.ve.kavachart.applet.scrollingLine) extends the basic functionality of lineApp to add a scrollbar at the bottom of the chart. This scrollbar is used to page through views of your data.

This applet adds a new parameter “scrollWindows” that specifies the number of pages that the data should be broken into. For example, if you set the number of “scrollWindows” to 5 and were viewing 1000 observations, spaced evenly along the X axis, each scroll page would show 200 points. The default number of scroll windows is 10.

Timeseries Applet Collection

This applet collection includes a variety of charts designed to display time series. These charts are typically used to help understand trends over time, and find extensive use in finance, transportation, energy, and technology.

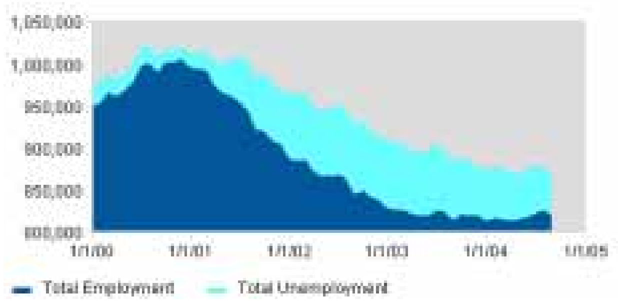
These applets use a specialized axis that automatically finds appropriate labeling increments based on the time range displayed, and uses localized conventions for things like day and month names, week start and ends, and so on. These time oriented axes are discussed in the previous chapter.

Also, these charts take data in a special way, using dataset0dateValues instead of dataset0xValues, and customDatasetHandler for block data handling. See the previous chapter for more information on timeseries data handling.

These applets also support specialized date formatting for dwell labels and axis labels. See the previous chapter for details.

Date Area Charts

Time-oriented area charts are terrific for showing trends over time.



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, dwell labels and hyperlinks are disabled for this type of chart.

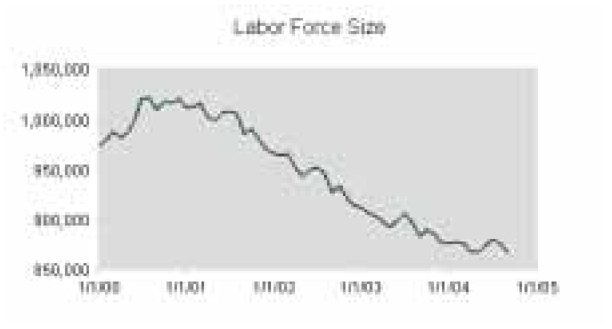
It's important for area charts with multiple datasets to use the same X values for every dataset. Otherwise the areas cannot stack properly.

Parameter	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)

Date Line Charts

These charts can be used as either scatter plots or as trend lines. One applet handles discontinuous data, creating line breaks when data is unavailable. Another lets you scroll through large datasets.

com.ve.kavachart.applet.dateLineApp

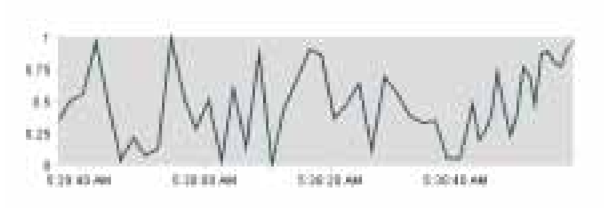


com.ve.kavachart.applet.scrollingDateLineApp
com.ve.kavachart.applet.disDateLineApp

Parameter	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).

Strip Charts

This chart is most appropriate for displaying continuously updating data. See the previous chapter for more information about reading data from a URL, and using "networkInterval" to specify how often data should be refreshed. A new parameter "maxNumberDataPoints" determines how many points the chart will contain. When the number of points exceeds this number, the earliest points will be deleted.

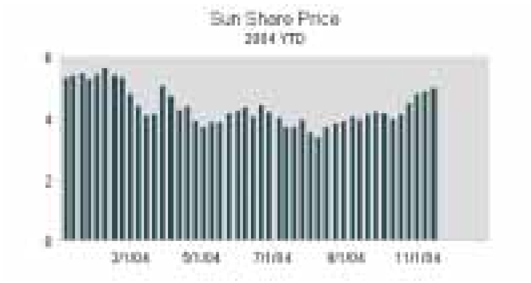


Unlike other timeseries line charts, data in strip charts always goes from one end of the plotarea to the next. Also, labels may not be displayed at the time axis ends, but is instead displayed at the correct location in appropriate increments.

Strip charts support other line chart parameters.

Time-oriented Bar Charts

Bar and Column charts are generally not the best vehicles for displaying time-series data, but they're available if you need them for your application. These charts should have at least 20 points to display appropriately, and will not have a label beneath each bar, but rather at appropriate time increments.



com.ve.kavachart.applet.dateBarApp

com.ve.kavachart.applet.dateColumnApp

com.ve.kavachart.applet.dateStackBarApp

com.ve.kavachart.applet.dateStackColumnApp

Bar charts have variable bar width, an adjustable baseline, and labels that can be toggled on or off. If you want each bar to have a different color, set the parameter “individualColors” to true, and define the colors with “dataset0Colors”.

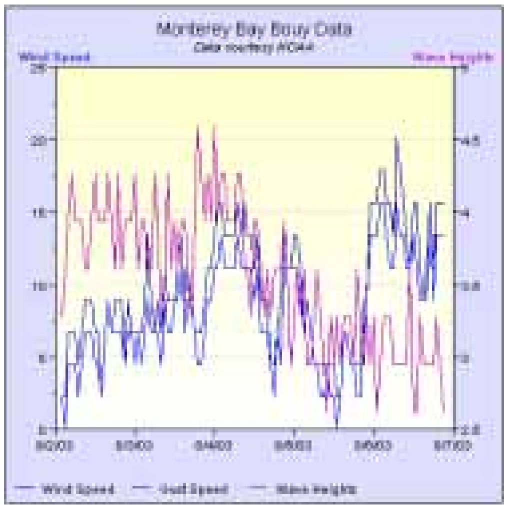
dateStackBarApp and dateStackColumnApp stack datasets instead of clustering them. This is useful to display a cumulative summary along with the individual data.

If you’re in a Java 2 or better environment, dataset image parameters will cause your bars to be drawn using tiles of the specified image.

Parameter	**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 the bar labels are the dataset labels or just the y values (default is false)

Multiple Axis Charts

Some 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 twinAxisDateLineApp to assign baseball scores the the right axis, and basketball scores to the left axis.



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

com.ve.kavachart.applet.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.

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.

Parameter	**value type**	**effect**
datasetNType	Line | Stick	This determines the DataRepresentation for datasetN. TwinAxisDateComboApp only
datasetNonRight	true|false	This determines whether dataset N will be assigned to the standard left axis or the auxilliary right axis.
plotLinesOn	anything	plot lines should display (default).
plotLinesOff	anything	Create a scatter plot by making plot lines invisible.
auxPlotLinesOn	anything	plot lines should display (default).
auxPlotLinesOff	anything	Create a scatter plot by making plot lines invisible
barBaseline	double	bars ascend or descend from this value, applicable to Sticks in twinAxisDateComboApp
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
auxBarBaseline	double	sets the baseline value for the Stick's assigned to the aux axis in TwinAxisDateComboApp
barWidth	integer	sets the width in pixels of the Stick's in TwinAxisDateComboApp
auxBarWidth	integer	sets the width in pixels for the Stick's assigned to the aux axis in TwinAxisDateComboApp

scrollingDateLine

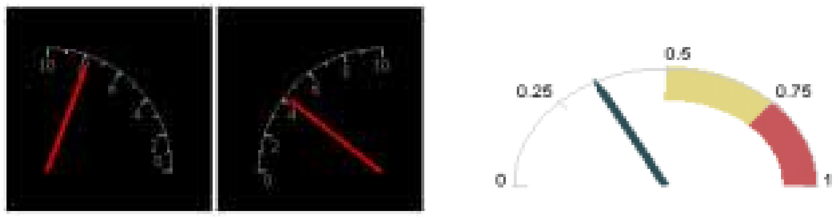
This applet extends the basic functionality of dateLineApp to add a scrollbar at the bottom of the chart. This scrollbar is used to page through views of your data.

A new parameter “scrollWindows” specifies the number of pages that the data should be broken into. For example, if you set the number of “scrollWindows” to 5 and were viewing 1000 observations, spaced evenly along the X axis, each scroll page would show 200 points. The default number of scroll windows is 10.

Specialty Applet Collection

Speedos

These include com.ve.kavachart.applet.speedoApp and com.ve.kavachart.applet.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 applet 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 each dataset. However, the other values are considered for building the speedo’s scale.

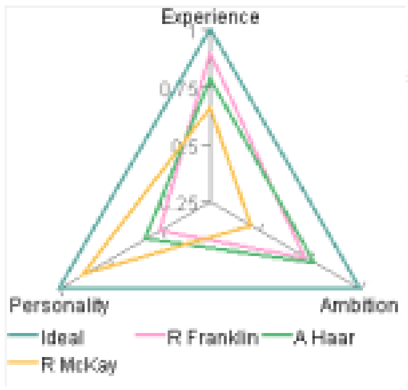
To get multiple speedos on a page, use multiple applet definitions.

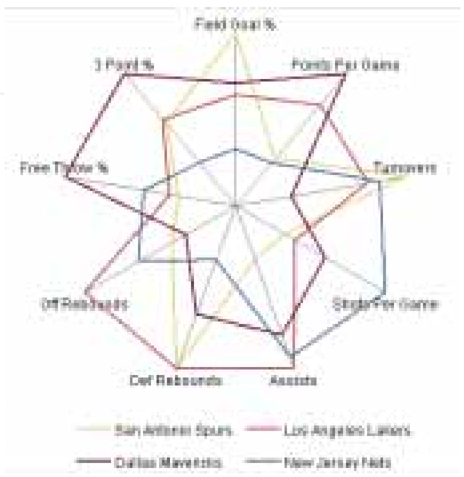
Speedo Chart Parameters	value type	effect
needleStyle	integer	Kind of needle (default 1) 0 = arrow, 1 = line, 2 = thick arrow, 3 = swept arc
speedoPosition	integer	0 (default) is a mostly complete circle, 1 - 4 are semi circles in various positions, 5-8 are quarter circles in various positions
labelsInside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

Kiviat Diagrams, Radar Plots, Polar Charts

KavaChart “polar charts” are sometimes 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.

com.ve.kavachart.applet.polarApp



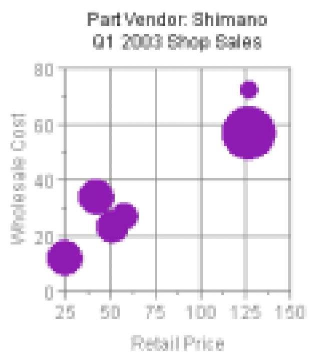


Polar Chart Parameters	value type	effect
manualSpoking	true|false	If defined, you are responsible for determining how many "spokes" should be drawn in this chart's axis representation
numSpokes	integer	The number of spokes in this chart's Axis system (default 4)
xAxisLabels	List	Labels for the edge of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

Bubble Charts

KavaChart bubble charts use the applet com.ve.kavachart.applet.bubbleApp. This chart draws circles at X,Y values specified by dataset0xValues and dataset0yValues. The size of the circle is determined by dataset0y2Values.

com.ve.kavachart.applet.bubbleApp



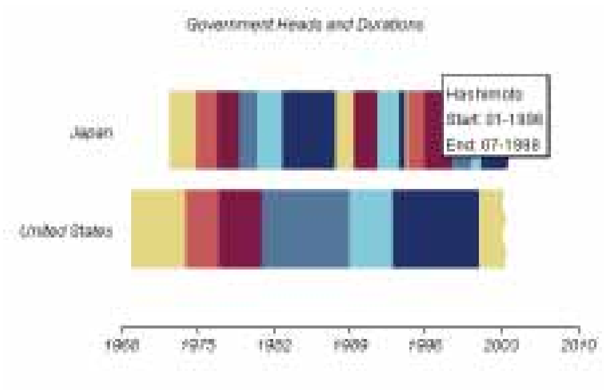
These charts may have filled or hollow circles, crossing X and Y axes, and manual or automatic Z scaling. Z scaling refers to the relative size of the bubbles, based on the overall set of Z (y2) values.

Bubble Chart Parameters	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

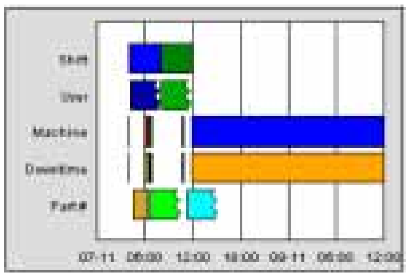
Gantt charts are a specialized chart designed to show when tasks start and end. This sort of chart is particularly useful for resource allocation and project planning, but it can also be used to visually describe the progress of multiple projects or processes.

com.ve.kavachart.applet.ganttApp



This applet 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 param “useTearEdge” is set to “true”.



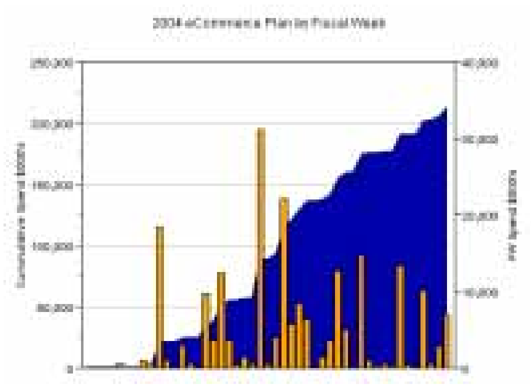
Another special param for this applet, minBarWidth, ensures that very narrow bars, like those in the applet above, will remain visible.

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. In an applet, this string should have a '#' character where the date will occur. Default is "Start #"
dwellEndString	String	This string defines the dwell label string for the end date. In an applet, this string should have a '#' character where the date will occur. Default is "End #"
dwellIndefiniteString	String	This string defines the dwell label string for an indefinite start/end. Default is "Indefinite"

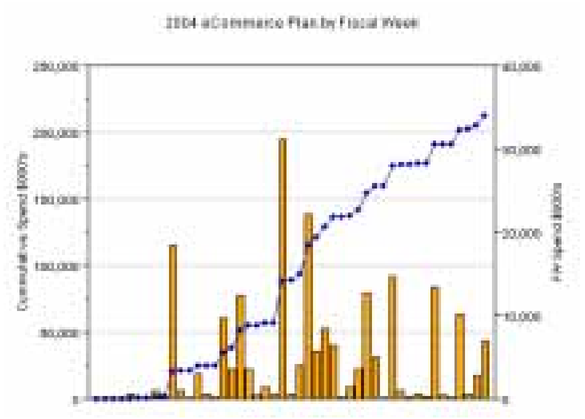
Multi-Axis Charts

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.

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



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



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

com.ve.kavachart.applet.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.

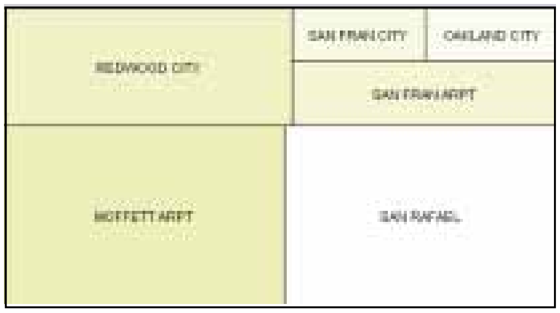
Parameter	value type	effect
datasetNType	Bar | Line | Area	This determines the DataRepresentation for datasetN. "Area" is only available for TwinAxisBarAreaApp, Line is not available for TwinAxisBarAreaApp, and so on. Stick is only available for twinAxisDateComboApp
datasetNonRight	true|false	This determines whether dataset N will be assigned to the standard left axis or the auxilliary right axis. Only applicable to twinAxisLineApp.

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 twinAxisLineApp.
auxPlotLinesOff	anything	Create a scatter plot by making plot lines invisible. Applicable to twinAxisLineApp.
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
areaBaseline	double	sets the baseline value for this area
auxBarBaseline	double	sets the baseline value for the Stick's assigned to the aux axis in TwinAxisDateComboApp
barWidth	integer	sets the width in pixels of the Stick's in TwinAxisDateComboApp
auxBarWidth	integer	sets the width in pixels for the Stick's assigned to the aux axis in TwinAxisDateComboApp

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.

com.ve.kavachart.applet.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.

This applet is particularly useful when coupled with URL hyperlinks to implement a drill-down facility.

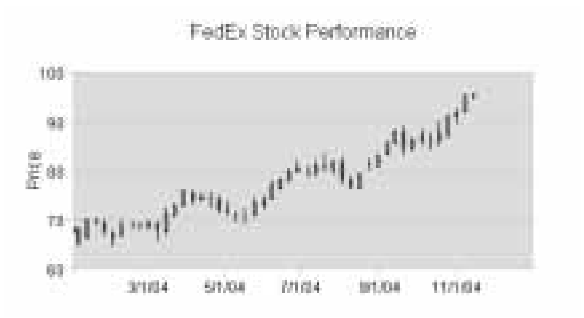
Parameter	value type	effect
individualColors	True|false	Determines whether colors should come from "dataset0Colors"
gradientColoring	True|false	Determines whether colors should be auto-graduated from the dataset color to the "secondary color"
sectorSecondaryColor	Color	A second color to be used for gradient 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.

Finance Applet Collection

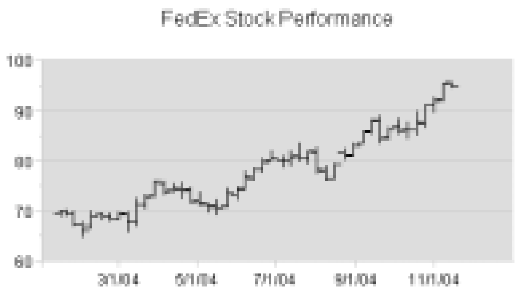
Candlestick and OHLC Charts

This collection includes some standard charts for dealing with financial data: com.ve.kavachart.applet.candlestickApp and com.ve.kavachart.applet.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.

com.ve.kavachart.applet.candlestickApp



com.ve.kavachart.applet.hLOCApp



Special parameters for these applets:

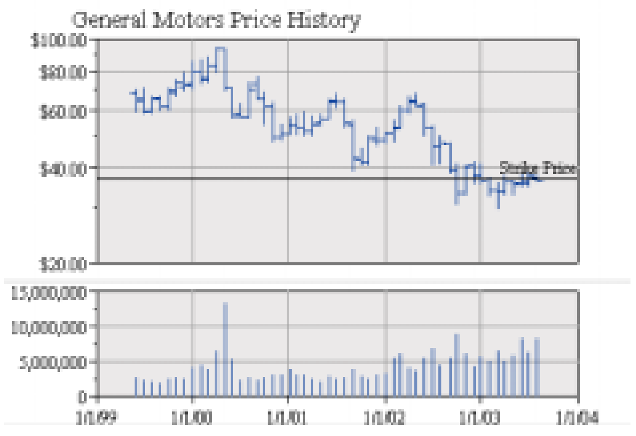
Parameter	value type	effect
dataset0highValues	list	High price at observed dates
dataset0lowValues	list	Low price at observed dates
dataset0openValues	list	Open price at observed dates
dataset0closeValues	list	Close price at observed dates
dataset0dateValues	list	List of dates in “inputDateFormat”
CustomDatasetHandler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

com.ve.kavachart.applet.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

com.ve.kavachart.applet.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

com.ve.kavachart.applet.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.

All these charts can 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.

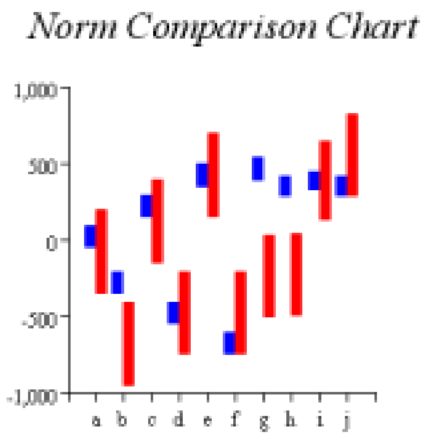
Parameter	value type	effect
datasetNType	HLOC|Stick|Line	dataset N will be either Stick, HLOC, or Line, based on this value. (finComboApp only).
splitWindow	true|false	if true (default) each dataset type will be in a a separate window with an independent Y axis. The X axis will be shared among all dataset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

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

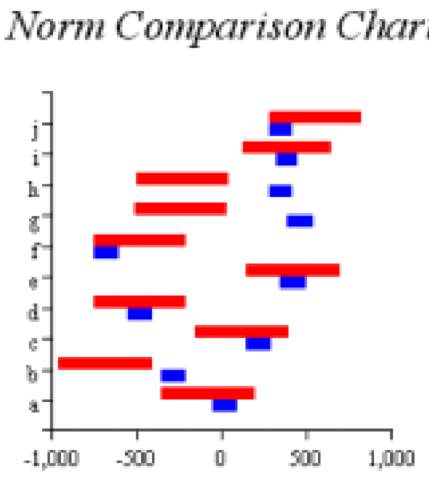
Hi-Lo Bar Charts

The finance package also includes hi-lo bar charts:

com.ve.kavachart.applet.hiLoBarApp



com.ve.kavachart.applet.hHiLoApp



Parameter	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 the bar labels are the dataset labels or just the y values (default is false)

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

It's also worth noting that there are also several finance-oriented applets and servlets in the com.ve.kavachart.contrib package, also part of the Enterprise Edition. These include charts that overlay markers on candlestick charts, box-jenkins statistical charts, histograms, and line charts with zooming, scrolling, and multiple windows. Also, with a little bit of Java programming, you can combine various KavaChart elements into an endless variety of custom finance charts. These elements are described in more detail in the KavaChart Enterprise Edition documentation.

Appendix

A

Obtaining and Using Your License Key

If you're using the demo download of KavaChart applets, you have undoubtedly noticed the demo behaviour. This section describes how to obtain and use a license key to eliminate this behavior.

Obtaining a License Key

Your KavaChart license keys can be obtained from our web site http://www.ve.com, under the "MyKavaChart" section. The license key is a simple text string that eliminates the demo behavior when used as a parameter.

Using a License Key

What do you do with a license key? Simply add it as a parameter to your applet definiton, like this:

```
<applet code="com.ve.kavachart.applet.columnApp" …
<param name="appletKey" value="XXXX-xxx">
<param name="titleString" value="hello world">
</applet>
```

If this fails to remove the demo notices from your charts, please contact support@ve.com to diagnose and repair the problem.

Index