**Chapter 5. Direct use of D3 library**

The Chart class contains some methods to execute any valid javascript.

void addExecutableJavascriptToRunOnce(String javascript) executes any valid javascript only once.

Sometimes it is necessary to repeat the execution of the javascript each time when the chart is redrawn (for example, when it is resized or new chart commands are executed).

For the purpose of repeated execution of javascript there are 3 methods to help it:

int addExecutableJavascript(String javascript)

which executes the javascript but also stores it in memory and will repeat its execution each time the chart is redrawn. The function returns a generated id, which can be useful to delete this javascript from the pool.

void cleanAllExecutableJavascript() - removes all javascript commands from the pool, added with the method addExecutableJavascript(String javascript).

If you need to remove not all javascript commands, but only some of them, there is a method as follows:

void deleteExecutableJavascript(int javascriptId)

where the javascriptId is an id generated by the method addExecutableJavascript(String javascript).

You can use the direct execution of D3 commands together with the chart functionality or without the chart functionality.

If you need to disable the chart functionality, that is, to use this library just as a d3 wrapper, call the method as follows:

chart = **new** Chart();

chart.setDisabledWholeChart(**true**);

Even if you disable the chart functionality, some elements are created, namely:

<div class=”v-chart”>

<svg>

<g class=”chart-data-box-low”></g>

<g class=”chart-data-box-medium”></g>

<g class=”chart-data-box-high”></g>

</svg>

</div>

They are also present if the chart functionality is enabled.

If you are going to directly execute d3 commands, we recommend you to adhere to the policy as follows:

1. Set the style name of the chart and refer to this chart by that class.

For example,

Chart chart = new Chart();

chart.addStyleName(“chart1”);

Then in the d3 command you can refer to this chart as follows: d3.select(“.chart1”);

1. You have to make a decision whether you need the redrawing support of this library or not

Advantages of redrawing:

1. The chart or picture is automatically resizable (if you follow this policy)
2. If you use the chart functionality, this is very recommended

Advantages of no redrawing:

1. Each redrawing takes time, so no redrawing works faster.
2. Less complexity if you add animation
3. SVG has some embedded methods of automatic resizing
4. If you decided do not use redrawing capabilities of this library, put all your graphic elements as follows:

d3.select(“.chart1 svg”).append(…)

where chart1 is the style name you added with chart.addStyleName(“chart1”)

To execute your javascript, use

Chart. addExecutableJavascriptToRunOnce(String javascript)

1. If you decided to use redrawing capabilities of this library, put all your graphic elements in one of 3 ways as follows:

d3.select(“.chart1 g.chart-data-box-low”).append(…)

or

d3.select(“.chart1 g.chart-data-box-medium”).append(…)

or

d3.select(“.chart1 g.chart-data-box-high”).append(…)

These are 3 layers and the first layer is the lowest.

When you add your graphic element, use the function

Chart. addExecutableJavascript(String javascript)

If you need to extend or reduce your image during resizing, use the following info:

window.getComputedStyle(d3.selectAll(“.chart1 svg”)[0][0],null).getPropertyValue(“width”)

is the current width of the chart svg area,

window.getComputedStyle(d3.selectAll(“.chart1 svg”)[0][0],null).getPropertyValue(“height”) is the current height of the chart svg area.

When resizing occurs, your javascript will be surely re-executed if you follow this policy. Before the re-execution, all old elements in g.chart-data-box-low, g.chart-data-box-medium, g.chart-data-box-high are completely removed by this library.

**EXAMPLE**

Now we will give you an example how to draw a pie with the text inside, which you might view in the Introduction.

We will draw it in 2 ways: with redrawing capability and without redrawing capability

1. WITH REDRAWING CAPABILITY

chart = **new** Chart();

chart.addStyleName("chart1");

chart.setWidth("100%");

chart.setHeight("500px");

chart.setDisabledWholeChart(**true**);

chart.addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low\").append(\"path\").attr(\"d\",\"M 300 230 l 0 -200 a 200 200 0 0 1 173.2 300 Z\").attr(\"fill\",\"blue\").attr(\"stroke\",\"#FFFFFF\").attr(\"stroke-width\",1).attr(\"stroke-linejoin\",\"round\")");

chart.addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low\").append(\"path\").attr(\"d\",\"M 300 230 l 173.2 100 a 200 200 0 0 1 -346.4 0 Z\").attr(\"fill\",\"red\").attr(\"stroke\",\"#FFFFFF\").attr(\"stroke-width\",1).attr(\"stroke-linejoin\",\"round\")");

chart.addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low\").append(\"path\").attr(\"d\",\"M 300 230 l -173.2 100 a 200 200 0 0 1 173.2 -300 Z\").attr(\"fill\",\"green\").attr(\"stroke\",\"#FFFFFF\").attr(\"stroke-width\",1).attr(\"stroke-linejoin\",\"round\")");

chart.addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low\").append(\"text\").text(\"Freedom\").attr(\"x\",290).attr(\"y\",330).attr(\"fill\",\"yellow\").attr(\"style\",\"font-size:14px\")");

chart.addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low\").append(\"text\").text(\"Faith\").attr(\"x\",400).attr(\"y\",190).attr(\"fill\",\"yellow\").attr(\"style\",\"font-size:14px\")");

chart.addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low\").append(\"text\").text(\"Moral\").attr(\"x\",190).attr(\"y\",190).attr(\"fill\",\"yellow\").attr(\"style\",\"font-size:14px\")");

1. WITHOUT REDRAWING CAPABILITY

Replace all entries of

addExecutableJavascript("d3.select(\".chart1 g.chart-data-box-low

with

addExecutableJavascriptToRunOnce("d3.select(\".chart1 svg

in the code above.

In the import section of the Java file add as follows:

**import** ua.net.freecode.chart.Chart;

**import** com.vaadin.Application;

**import** com.vaadin.ui.\*;

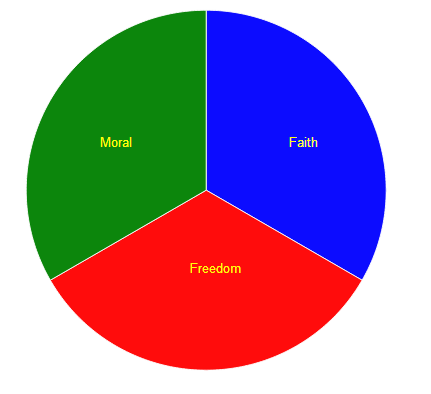
In the section of variables, add the declaration as follows:

**private** Chart chart;

Finally add the chart component to any Vaadin panel, layout or window, for example:

panel.addComponent(chart);

As a result, you will get the picture as follows:



You can find the full source code in the archive you downloaded in the folders: sample2forvaadin6 for Vaadin6 and sample2forvaadin7 for Vaadin 7.

For the sample 2, use the instructions as follows to run this sample:

Instructions for Vaadin6 sample 2 application

1. Create a new Vaadin 6 project in Eclipse

2. Replace your project's src folder with this src folder

3. Replace your project's WebContent/WEB-INF/web.xml file

with this web.xml file

4. Copy

chartlib/charts/d3.v3.min.js to <YOUR PROJECT>/WebContent/VAADIN/charts/d3.v3.min.js

chartlib/charts/chart\_1.0.js to <YOUR PROJECT>/WebContent/VAADIN/charts/chart\_1.0.js

chartlib/charts/fchart.css to <YOUR PROJECT>/WebContent/VAADIN/charts/fchart.css

chartlib/libforvaadin6/fchart-1.0.jar to <YOUR PROJECT>/WebContent/WEB-INF/lib/fchart-1.0.jar

Note: if any folder does not exist in your project, create it!

5. Refresh the project in Eclipse (left click on the project and press F5)

6. Specify that fchart-1.0.jar is one of required libraries:

Right-click on the project in Eclipse -> Properties -> Java Build Path

-> Libraries -> Add JARs -> find fchart-1.0.jar in your project tree

-> Ok

Also in Java Build Path -> Order and Export -> check fchart-1.0.jar

7. Compile your java code (for example, Project -> Clean -> your project,

check immediate rebuild)

Note: if you have the error: internal validation error for d3.v3.min.js, see Trouble-Shooting Guide to solve this problem.

8. Compile Vaadin widgets (Press button in Eclipse: Compile Vaadin Widgets)

9. Run the application on the server (for example, on Tomcat)

Instructions for Vaadin7 sample 2 application

1. Create a new Vaadin 7 project in Eclipse

2. Replace your project's src folder with this src folder

3. Copy

chartlib/libforvaadin7/fchart-1.0.jar to <YOUR PROJECT>/WebContent/WEB-INF/lib/fchart-1.0.jar

chartlib/libforvaadin7/fchart-1.0.src.jar to <YOUR PROJECT>/WebContent/WEB-INF/lib/fchart-1.0.src.jar

Note: if any folder does not exist in your project, create it!

4. Refresh the project in Eclipse (left click on the project and press F5)

5. Specify that fchart-1.0.jar is one of required libraries:

Right-click on the project in Eclipse -> Properties -> Java Build Path

-> Libraries -> Add JARs -> find fchart-1.0.jar in your project tree

-> Ok

Also in Java Build Path -> Order and Export -> check fchart-1.0.jar

6. Specify that fchart-1.0.src.jar is one of required libraries:

Right-click on the project in Eclipse -> Properties -> Java Build Path

-> Libraries -> Add JARs -> find fchart-1.0.src.jar in your project tree

-> Ok

Leave fchart-1.0.src.jar unchecked in Order and Export

Note: fchart-1.0.src.jar is necessary only for widget compilation

and as a source code for your pleasure.

7. Compile your java code (for example, Project -> Clean -> your project,

check immediate rebuild)

Note: if you have the error: internal validation error for d3.v3.min.js, see Trouble-Shooting Guide to solve this problem..

8. Compile Vaadin widgets (Press button in Eclipse: Compile Vaadin Widgets)

9. Run the application on the server (for example, on Tomcat)

Note: the Eclipse's embedded browser does not show the chart correctly,

use any modern browser.

God bless you!