University of Rouen

Master 1 GIL - 2010/2011

User Manual - HTML5Charts

Project Team
Pierre Collignon
Ludovic Thueux
Maxence Luce
Lyes Kimouche
Radu Popica
Abdourahmane Djigo

Client
Florent Nicart

Contents

Chapter 1

Product description - A chart library in HTML5

The *HTML5Charts* product is a JavaScript library allowing its users to draw charts from a coherent data set. The automated data treatment used to display is entirely customisable. This is done via the options given to the user, or via reimplementations of the javascript interfaces made for this purpose.

This product has been built to prove that HTML5 is able to progressively replace Flash.

1.1 The diagrams

Four types of diagrams are available :

- The bar diagram
- The 3D bar diagram
- The pie chart
- The line diagram

For each of these components, two views can be drawn since the data are stored in a double entry array model.
We can therefore draw using the rows or the columns of this array as a reference.
The drawings will be differents and the choice between the two of them is done by the user according to the data he would like to focus on.
An example of this concept could be a society willing to display a chart containing the results of a survey on the number of buyers for each product it sells and for each of its branches. The user may want to either draw the number of buyers in each branch for each product, or the the number of buyers of each product by branch.

The library is used by two types of users: the user willing to insert a chart in his website and the user reading the chart on this website. They will be called the ”developper” and the ”end user”.

The developper's goal, as said previously, is to insert a chart in a website.
Il will communicate with the library using an XML-formatted file respecting constraints given by an XSD file provided with the project source code. This data communication can happen in two ways: either writing the data inside the HTML page or receiving the data from a third party data provider.
The style - the colors used by the drawings, the legend position - can be provided exactly as the data - either internally or externally.
Every combination is possible, the data can be provided internally while the style is provided externally and vice versa.

For the end user the chart is already displayed on the webpage he's visiting. But he's also able to interact with the charts - except for the line diagram. When the end user moves its mouse cursor on one of the chart's division - either a bar for the bar diagram or a slice for the pie chart - the division is highlighted and its value displayed in a tooltip text.

Chapter 2

Using the library - for beginners

The library can be used immediately without knowing the complex mechanisms lying under the hood in order to display charts using the internal source with the default style.

2.1 Internal data source creation

The internal data source is an XML file contained in a `<pre>` html tag.
Nothing is required to use it, except making sure the XSD is respected by the XML file.
Once the data is described as an XML file, an `InternalDataSource` Javascript Object has to be used to read its content.
This object implements the `IDataSource` interface containing two methods: `lodaData(callback)` and `getDataMatrix()` .
The `loadData` method takes as parameter a callback function called when the data are loaded while the `getDataMatrix` method enables the user to access an abstract view of the data, usable during the whole life of the program.

2.2 Chart creation

Four chart classes are available: `HistoDiagram`,`Histo3DDiagram` ,`PieDiagram` and `LineDiagram`.
The chart creation happens after the data source loading by giving to the desired class' constructor the DOM object corresponding to the Canvas object inside of which the user wants the chart to be drawn as well as the type of data focus he wants (either 'row' or 'column'). Then, once the object is created, we can use the `setData(dataMatrix)` method to provide the chart with a dataMatrix.
The `setData` method redraws the content immediately.

2.3 Example

We're considering that the `<pre>` tag containing the xml file has `pre1` as identifier and that the `<canvas>` tag containing the drawing area has `canvas1` as identifier.

```
var ids = new InternalDataSource('pre1');
ids.loadData(function() {
// As soon as the data source is loaded, this function is called
   var diag = new PieDiagram(document.getElementsByTagName('canvas1')[0], 'row');
   diag.setData(ids.getDataMatrix());
});
```

Chapter 3

Using the library -for advanced users

We're explaining here a more complex way of using the library, enabling one to use external XML files as well other file format, and to change the default style used for the chart drawing.

3.1 External sources usage

The external sources let the user deleguate the XML file providing to a third party provider, who will still need to respect the XSD specification. The usage is similar to the internal source creation, except this time you have to create an `ExternalDataSource` object and give him the XML file's url as a parameter instead of the `<pre>` identifier.

3.2 Connectors

The connectors are one of the concepts enabling the modularity of the library. These connectors must be able to create a chart for any type of data. The user can, for example, create a chart and use the JSON format instead of the XML format.
A connector is created by the developper by providing the `ConnectorDataSource` the custom implementation of the `loadData` and `getDataMatrix` functions .

An example using the JSON format is provided in the project source code.

3.3 Applying Style

Changing style parameters allows the developper to deeply change the chart's appearance. In order to stay coherent with the way we provide data, the stylisation process follows the same rules: an XML file describing the style, either internal or external, is given to the library.

3.3.1 The stylisation XML file

The stylisation consists of providing an alternative color set as well as positionning parameters for the legend.

```
<style>
<colors>
<color>Color_name</color>
...
<colors>
<legend>
<x>x coordinate of the top-left corner</x>
<y>y coordinate of the top-left corner</y>
<w>rectangle width</w>
<h>rectangle height</h>
</legend>
</style>
```

3.3.2 Internal and External Style Source

The way the style sources are used is very similar to the way the data sources are used:
The user first creates an `InternalStyleSource` or an `ExternalStyleSource` by providing either a `<pre>` tag identifier or and XML file url. Both of the available methods in these objects implementing the `IStyleSource` interface are `loadData` and `getStyleSource` which behave exactly as their `IDataSource` equivalents.