

# HyJavaCharts Full Documentation

the java api for highcharts

## OVERVIEW

The HyJavaCharts Java library is a pure Java wrapper for the Highcharts JavaScript library. It allows your Java-based web applications to configure Highcharts charts using only Java methods. There is no need to write any JavaScript. The library will generate the required JavaScript code that Highcharts needs to render the charts.

The Highchart class is the starting point for all charts.

To define the contents of a chart, you simply have to instantiate the Highchart class and use its setter methods to define the chart options and chart data as required. For details on the effects of each chart option refer to the [Highcharts API](#) reference. The naming of the chart options in the HyJavaCharts library in most cases exactly follows the Highcharts JavaScript API.

In the example below we are defining a bar chart. Instantiate the Highchart class and then get a reference to ChartOptions. You then set the chart options as required.

```
public class YourBarChart {

    public Highchart configure() {
        Highchart highChart = new Highchart();
        ChartOptions chartOptions = highChart.getChartOptions();

        chartOptions.getChart().setType(ChartType.BAR);
        chartOptions.getTitle().setText("Historic World Population by Region");
        chartOptions.getSubTitle().setText("Source: Wikipedia.org");

        set all your required chart options.....

        return highChart;
    }

}
```

## JAVADOC API

In most cases the HyJavaCharts API exactly matches the Highcharts API. There is no need to learn another API. The [HyJavaCharts API](#) can be viewed online or refer to the API jar provided in the product download.

## DEMO WEB APPLICATION

To give you a kick start using the library, the demo web application provides thorough examples of more than 100 different charts including Java source code for each example. Most of the Highcharts Demos from the [Highcharts Demos](#) page are included in this application.

The Java source code for the demo application including all chart examples is available from the [Other Downloads](#) page.

## HOW TO USE THE LIBRARY

### Java

Include the HyJavaCharts jar in your project.

Configure your chart and generate the JavaScript code for the global and chart options.

```
Highchart hc = new YourBarChart().configure();
String globalOptionsJs = hc.globalOptionsToJs();
String chartOptionsJs = hc.chartOptionsToJs();
```

The YourBarChart class will look something like this:

```
public class YourBarChart {

    public Highchart configure() {
        Highchart highChart = new Highchart();
        ChartOptions chartOptions = highChart.getChartOptions();

        chartOptions.getChart().setType(ChartType.BAR);
        chartOptions.getTitle().setText("Historic World Population by Region");
        chartOptions.getSubTitle().setText("Source: Wikipedia.org");

        set all your required chart options.....

        return highChart;
    }

}
```

### JavaScript

Include required JavaScript on your page as per Highcharts documentation.

```
Highcharts.setOptions({globalOptionsJs});
Highcharts.chart('container', ${chartOptionsJs});
```

Your Java code will pass the globalOptionsJs and chartOptionsJs strings to the markup variables \${globalOptionsJs} and \${chartOptionsJs}. The method used to pass the generated chart options from the Java to the web page will vary depending on your Java web framework. (Refer to *Java Web Frameworks below*).

### Web Page

Your markup will look something like this:

```
<script src=https://code.highcharts.com/highcharts.js></script>

<div id="container" style="min-width: 310px; max-width: 800px; height: 400px; margin: 0 auto"></div>

<script>
Highcharts.setOptions(${globalOptionsJs});
Highcharts.chart('container', ${chartOptionsJs});
</script>
```

That is all that is required to get a working Bar chart example.

## MAIN LIBRARY CLASSES

### GlobalOptions Class

The GlobalOptions class maps directly to the Highcharts global options in the Highcharts API.

All Highchart global options will be configured using setter methods on this class.

### ChartOptions Class

The ChartOptions class maps directly to the Highcharts chart options in the Highcharts API.

All Highchart chart options will be configured using setter methods on this class.

### Highchart Class

The Highchart class wraps both the GlobalOptions and ChartOptions classes into one convenient class to make usage as simple as possible.

This class handles chart theming; the setting of global and chart options; generates JavaScript for GlobalOptions and ChartOptions.

All Highchart global options can be configured by getting a reference to GlobalOptions from this class via getGlobalOptions().

All Highchart chart options can be configured by getting a reference to ChartOptions from this class via getChartOptions().

Usage of the Highchart class has been described in examples above.

### HighchartOptions Class

As an alternative to using the Highchart class, global and chart options can be configured separately by using the GlobalOptions and ChartOptions classes outside of the Highchart class. Global options can be configured once and will apply to multiple charts. Global options are optional.

Usage is described below.

The HighchartOptions class handles chart theming and generates JavaScript for GlobalOptions and ChartOptions.

Configure your chart and get the generated JavaScript code for the global and chart options.

```
GlobalOptions globalOptions = new GlobalOptions();
globalOptions.getGlobal().setUseUTC(false);
set all your required global options.....

ChartOptions barChartOptions = new YourBarChart().configure();

String globalOptionsJs = new HighchartOptions().toJs(globalOptions);
String chartOptionsJs = new HighchartOptions().toJs(barChartOptions);
```

The YourBarChart class will look something like this:

```
public class YourBarChart {

    public ChartOptions configure() {
        ChartOptions chartOptions = new ChartOptions();

        chartOptions.getChart().setType(ChartType.BAR);
        chartOptions.getTitle().setText("Historic World Population by Region");
        chartOptions.getSubTitle().setText("Source: Wikipedia.org");

        set all your required chart options.....

        return chartOptions;
    }

}
```

### HighchartRenderer Class

The HyJavaCharts library performance is extremely fast right out of the box.

In cases where thousands of charts are generated, for example batch processes, some improvements can be found using HighchartRenderer.toJs(...) rather than the Highcharts.chartOptionsToJs() and Highcharts.globalOptionsToJs().

Instantiate the HighchartRenderer class once and then use it to generate the chart options JavaScript for multiple charts.

Note that there are no improvements when generating single charts.

## EXTENDED GLOBAL OPTIONS

The Highcharts documentation also discusses the ability to use the Highcharts.setOptions(...) to configure various options that are not described in the setOptions API.

For Example: A global font family can be set using the chart.style option.

```
Highcharts.setOptions({
  chart: {
    style: {
      fontFamily: 'serif'
    }
  }
});
```

To do this using the HyJavaCharts library, you could define a MyGlobalOptions class as below.

```
public class MyGlobalOptions {

    public Highchart configure() {
        Highchart highChart = new Highchart();

        ChartOptions chartOptions = highChart.getChartOptions();
        chartOptions.getChart().getStyle().setFontFamily("serif");

        return highChart;
    }

}
```

Configure and generate the global options JavaScript and then use the myGlobalOptionsJs variable in the Highcharts.setOptions(...).

```
Highchart myGlobalOptions = new MyGlobalOptions().configure();
String myGlobalOptionsJs = myGlobalOptions.chartOptionsToJs();
```

If you need to also define global or lang options, you can use Highcharts.setOptions(...) multiple times.

Of course you could also achieve the same thing by defining a super class for all charts and setting the fontFamily in that class.

## OPTIONS WITH MULTIPLE DATA TYPES

Where a chart option has more than one data type, eg. Number and String the Java API will include one method per type. Setter methods will be overloaded. Getter methods will include a suffix.

```
chartOptions.getChart().setWidth(25);
chartOptions.getChart().setWidth("50%");

chartOptions.getChart().getWidthAsNumber();
chartOptions.getChart().getWidthAsString();
```

## SETTING OPTIONS TO NULL

In some (rare) cases you need to set a Highcharts chart option to null. Where a chart option has only one datatype you simply set the option to null as below.

```
chartOptions.getChart().setBorderRadius(null);
```

Where a chart option has more than one data type, eg. Number and String, the setter method will be overloaded. You can use the helper class NullOption to set the option to null.

```
chartOptions.getChart().setWidth(NullOption.NULL_STRING);
OR
chartOptions.getChart().setWidth(NullOption.NULL_NUMBER);
```

Where a chart option has more than one data type and the type is a chart option class, eg. Chart.animation has datatypes Boolean and AnimationOptions. You can use the helper static variable AnimationOptions.NULL to set the option to null. Every chart option class has a static variable named NULL to make this as easy as possible.

```
chartOptions.getChart().setAnimation(AnimationOptions.NULL);
```

## MAIN OPTION DATA TYPES

### Color Class

Maps to the Highcharts Color, Highcharts.ColorString, Highcharts.GradientColorObject, Highcharts.PatternOptionsObject data types.

To set a color string value use: new Color("#55BF3B"); or new Color().setColor("#55BF3B");

To set a JavaScript color value use: new Color().setColorValue("jsvalue");

To set a RGB color value use: new Color(255, 255, 255); or new Color().setRGB(255, 255, 255);

To set a RGBA color value use: new Color(255, 255, 255, 0.5); or new Color().setRGBA(255, 255, 255, 0.5);

To set a linear or radial gradient color value use: new Color(LinearGradient, stops); or new Color(RadialGradient, stops);

To set a pattern fill color value use: new Color(PatternOptions);

To brighten a color use: new Color("#55BF3B").brighten(0.2); or new Color(255, 255, 255).brighten(0.2); or new Color(255, 255, 255, 0.5).brighten(0.2);

140 Web color code strings are also defined in this class. Eg. new Color(Color.ALICEBLUE).

### CSSObject Class

Maps to the Highcharts CSSObject or Highcharts.CSSObject data type.

To set a CSS JavaScript value use: new CSSObject("jsvalue"); or new CSSObject().setCssValue("jsvalue");

Use the methods provided to set any of the other available CSS options.

### Function Class

Maps to the Highcharts Function or any Highcharts.\*CallbackFunction data type.

To set the body of a Function use: new Function("functionbodystring"); or new Function().setFunctionBody("functionbodystring");

To add parameters to the function use: new Function("functionbodystring", "parameter"); or addParameter("parameter");

This class also provides a method setEmitAsString(boolean emitAsString) to emit this function as a string in the generated JSON. See the JSON section below for more details.

### General Class

To set a JavaScript value use: new General("jsvalue"); or new General().setJsValue("jsValue");

## JSON

Highcharts call back functions like "formatter" that are inline functions are not strict JSON. JSON does not allow inline functions.

Some Highcharts Color values, for example **new Highcharts.Color("#FFFFFF").brighten(0.2).get('rgb')**, are also not strict JSON.

If you use Highcharts.getJSON() in your markup or serialise/deserialise in your Java you may get an invalid JSON error.

The library gives you the option to treat the inline functions and Color values as strings. They can then be accepted by getJSON() and serialised/deserialised.

As Highcharts expects a function or a valid Color value, you will need to convert them back in your JavaScript before rendering the chart.

The HyJavaCharts library provides a JavaScript function: *hyJavaChartsInflateFunctions(yourChartConfig)* to zip them.

The JavaScript can be downloaded from <https://www.hyjavacharts.com/content/js/hyjavacharts-inflate-1.1.0.js> and included in your markup.

The Java class provides a method setEmitAsString(boolean emitAsString) – will emit this function as a string from the Highchart.chartOptionsToJs() method.

The Color class provides a method setEmitAsString(boolean emitAsString) – will emit this Color value as a string from the Highchart.chartOptionsToJs() method.

The Highchart class provides a static method setEmitFunctionsAsString(boolean emitFunctionsAsString) – will emit all functions as strings from the Highchart.chartOptionsToJs() method.

The Highchart class provides a static method setEmitColorsAsString(boolean emitColorsAsString) – will emit all Color values as strings from the Highchart.chartOptionsToJs() method.

An example using getJSON would look something like this:

```
<script src=https://.../hyjavacharts-inflate-1.1.0.js></script>

<script>
Highcharts.getJSON('https://...', function (chartoptions) {
  hyJavaChartsInflateFunctions(chartoptions);
  Highcharts.chart('container', chartoptions);
});
</script>
```

## ARRAYS

There are methods for each chart option array to help simplify usage where there is only 1 element in the array.

In the example the xAxisSingle() method will add a new XAxis object to the xAxis array and operate on that object.

```
XAxis xAxis = new XAxis();
xAxis.setAllowDecimals(false);
chartOptions.getXAxis().add(xAxis);
OR
chartOptions.getXAxisSingle().setAllowDecimals(false);
```

## THEMES

Include your chosen theme.js on your page as per Highcharts documentation. If your Java code uses the chartOptions.colors() or chartOptions.getColors() methods you will also need to set the Theme in your Java prior to calling those methods.

eg. *highChart.setTheme(highchartTheme.DARK\_UNICA)*;

All 10 standard Highcharts themes have been created in HighchartTheme.

If you have created your own theme you will need to create a Theme class as follows:

(1) create a new theme class implementing Theme.

(2) add the method getColors() to the class returning your theme colors.

```
public class YourNewTheme implements Theme {
    public List<Color> getColors() {
        return new ArrayList<Arrays.asList(
            new Color(Color.LIGHTSKYBLUE), new Color(Color.LIGHTGREEN), new Color(Color.LIGHTSALMON),
            new Color(Color.LIGHTCORAL), new Color(Color.LIGHTSTEELBLUE), new Color(Color.LIGHTGRAY),
            new Color(Color.LIGHTPINK), new Color(Color.LIGHTSEAGREEN), new Color(Color.LIGHTCYAN),
            new Color(Color.LIGHTBLUE), new Color(Color.LIGHTYELLOW), new Color(Color.LIGHTSLATEGRAY)
        ));
    }
}
```

You can also use color strings ie. new Color("#87cefa"). To use the theme: *highChart.setTheme(new YourNewTheme());*

If you are using the ChartOptions class then set your theme with: *new HighchartOptions().setTheme(HighchartTheme.DARK\_UNICA, chartOptions);* or *new HighchartOptions().setTheme(new YourNewTheme(), chartOptions);*

As an alternative to defining your theme in JavaScript it is also possible to define the full Theme in Java. Instantiate the Highchart class, get a reference to ChartOptions and configure all required theme chart options. You would then apply the theme on your web page by using the Highcharts.setOptions() JavaScript method. Please refer to the [Highcharts Theme](#) documentation for further details.

## CHART DATA SOURCES

Chart data sources can be databases, web services, files or any resource your application server has access to. The Highcharts Data module allows access to CSV, HTML tables or remote CSV data by specifying a URL.

## REQUIREMENTS

To render Highcharts charts, a commercial or free Non-commercial license **is required** for the Highcharts JavaScript library.

Please refer to the Highcharts [license](#) page for further details.

Note that usage of the actual HyJavaCharts library does not require a Highcharts license.

## JAVA VERSION

The HyJavaCharts library requires Java 8 or greater. There are no external dependencies.

## API ISSUES

Every endeavor has been made to ensure the HyJavaCharts Java API matches the Highcharts JavaScript API. Please contact us if there is a chart option missing from the Java API or if an option data type does not match the Highcharts API. We will publish an updated API as soon as possible.

## Generic Chart Options

To allow the developer to continue until the Java API is updated, the method *set(String name, Object value)* is available on every Highchart model class. This method allows you to configure any name/value pairs.

```
// If an option "allowAnimation" Boolean was missing from Chart in the API or the data type is incorrect
// you can still include the option as shown below
chartOptions.getChart().set("allowAnimation", true);

// If the option missing is a JSON object, you can use the General class.
chartOptions.getChart().set("someobject", new General("{ ... , ... }"));
```

## SUPPORT

For any questions or issues on the HyJavaCharts API, please contact our support team via the [Contact Us](#) area on the home page. Please refer to the [Highcharts API](#) for usage of each chart option or contact Highsoft for support of the Highcharts product.