Template:Highcharts

Documentation

This box includes basic usage information for this template. When calling the template, this documentation will not appear. Functional template code should be placed *outside* the dotted box.

Summary

Template Description	This is a wrapper for highly awesome Highcharts graphing package. Two reasons this package is great: 1. Produces beautiful graphs. 2. Based on JavaScript, SVG, and VML, so no plugins required and no dependency on Excanvas for IE compatibility. Has proven reliable on a wide variety of browsers. 3. Flexibility. You can make these charts do just about anything. Although the Highcharts package is quite complex, this template is designed to make it very easy to produce a wide variety of pleasing and useful graphs, by applying a reasonably intelligent set of defaults and interpretation of a small set of parameters. It also gives you access to (almost) the complete functionality of Highcharts should you desire it, in convenient DekiScript form.
Requirements	• Mindtouch 10.0. • **Highcharts is not free for commercial use, so please check licensing terms.** The fees are reasonable, help support worthy software! • You must have UNSAFECONTENT permission to install this template! • **You must attach the highcharts.js file to the template page on your wiki.**
Documentation URL	http://developer.mindtouch.com/App_Catalog/Highcharts_-_a_wrapper_for_the_Highcharts_JavaScript_graphing_package
Discussion URL	http://forums.developer.mindtouch.com/showthread.php?p=43365

Version History

Place newest version at the *top* of the table.

Version	Date	Author	Description
0.2.1b	15-Oct-2010	Neil Weinstock	Fixed a major bug in the handling of the **options** parameter.
0.2.0b	4-Oct-2010	Neil Weinstock	Added **labelFormatter**, **tooltipFormatter** and **categories** parameters. Upgrading to "beta".
0.1.0a	29-Sep-2010	Neil Weinstock	Basic functionality in place. Still evolving rapidly.
0.0.0a	28-Sep-2010	Neil Weinstock	Just getting started

Template Parameters

Name	Type	Default	Description
CORE	The following arguments are all you need to do anything you want. These can all be accessed either in an ordered list (`highcharts(type, size, data, options)`) or via an argument map (`highcharts { type:"line", etc. }`).		
type	str?	"line"	Default chart type. May be one of the following (case insensitive): • line • spline • area • areaspline • bar • column • scatter • pie • raw For all options except "raw", this template will generate a set of pleasing defaults for that chart type. For "raw", this template generate "renderTo" and

Name	Type	Default	Description
			"series" (from the **data** parameter), but leave everything else up to you to implement via the **options** parameter. In other words, the template will pretty much get out of the way and let you feed Highcharts exactly the options you want.
size	list of two numbers?	[400,200]	Width and height (in pixels) of chart container
data	list	*required*	List of data series. Format of each data series is chart type-dependent. **(add more here)**
options	map?	{}	Additional options passed directly to Highcharts. This options map is intelligently (?) merged with the options generated by the template based on the previous parameters. So you only need to specify options here to either override or modify what the template already provides. Note that label and tooltip formatters (if desired) *must* be specified using the dedicated options given below.
labelFormatter	str?	nil	Code for a custom plotOptions.series.dataLabels.formatter function. This lets you specify the exact contents of the labels shown on the chart for each data point. This is generally most useful for pie charts. See the linked documentation for explanation and examples. Note that the template will automatically generate the function wrapper; you only need to specify the

Name	Type	Default	Description
			contents of the formatter function (i.e., everything inside the {}).
tooltipFormatter	str?	nil	Code for a custom tooltip.formatter function. This lets you specify the exact contents of the tooltip shown when you mouse-over a point on the chart. This works will all chart types. As with **labelFormatter**, you only specify the contents of the function, and the template will generate the function wrapper.
debug	bool?	false	Show debug output. Specifically, this will display the argument map that is passed to Highcharts. This option requires the PrettyPrint template, and CollapseItem is recommended as well.
CONVENIENCE	The following arguments are provided as convenient shortcuts for applying some common modifications to the graph. These can be accessed *only* as named arguments in an argument map (`highcharts {}`).		
animation	bool?	false	Enable or disable animation when the graph is first displayed. Shorthand for **options**.plotOptions.series.animation.
categories	list?	nil	Category names for the x axis. Size of this list should be equal (at least) to the size of the data series. Shorthand for **options**.xAxis.categories.

Name	Type	Default	Description
title	str?	nil	Title text for the chart. This will be placed above the chart, centered. For more precise control, feed the required info directly to the **options** parameter. Shorthand for **options**.title.text, plus some additional interpretation.
stacking	str?	nil	Set to "normal" to stack based on summing the series. Set to "percent" to stack based on percentage of total. Shorthand for **options**.plotOptions.series.stacking.

Template Code

Unsafecontent Permission Check

```
// This code checks if the template is properly installed for unsafe cont
// and may be removed if this check is not desired.  If you leave this he
// in a new DekiScript block below this one.
var thisTemplate = wiki.inclusions()[-1];
if (!wiki.pagepermissions(thisTemplate.id, thisTemplate.author.id).unsafe
  <div style="color:red; width:75%; padding:5px; border:1px solid red;">
    "WARNING: The page '"..thisTemplate.path.."' must be re-saved by a us
    <a href="http://developer.mindtouch.com/en/kb/Using_templates_which_r
    " for more info.";
  </div>;
```

Parameter Processing

```
// Errors array.  There's lots of error checking to do, even if we can't
var errors = [];
var fatal = false;

// Convenience variables for all the different series types
var line = "line";
```

```
var spline = "spline";
var area = "area";
var areaspline = "areaspline";
var column = "column";
var bar = "bar";
var pie = "pie";
var scatter = "scatter";
var raw = "raw";

// ====================
//  Process Parameters
// ====================

// == Core Parameters ==

// type: what kind of chart (OPTIONAL; default "line")
var validTypes = [ line,spline,area,areaspline,column,bar,pie,scatter,raw
var type = $0 ?? $type ?? line;
if (type is str) let type = string.tolower(type);
if (!list.contains(validTypes, type)) {
    let errors ..= [ ("TYPE parameter must be one of "; validTypes) ];
}

// size: dimensions of chart area, in pixels (OPTIONAL; default 400x200)
var size = $1 ?? $size ?? [ 400, 200 ];
if (size is not list || size[0] is not num || size[1] is not num) {
    let errors ..= [ "SIZE parameter must be a list of two numbers" ];
}

// data: chart data.  This is always a list, with each list element a dat
var data = $2 ?? $data;
if (data is not list) {
    let errors ..= [ "FATAL: DATA parameter must be a list of data series
    let fatal = true;
}

// options: all other options for the chart (OPTIONAL)
var options = $3 ?? $options ?? {};
if (options is not map) {
    let errors  ..= [ "OPTIONS parameter must be a map" ];
}

// data label formatter
var labelFormatter = $4 ?? $labelFormatter;
if (labelFormatter is not nil && labelFormatter is not str) {
```

```
    let errors ..= [ "LABELFORMATTER parameter must be a string" ];
    let labelFormatter = nil;
}

// tooltip formatter
var tooltipFormatter = $5 ?? $tooltipFormatter;
if (tooltipFormatter is not nil && tooltipFormatter is not str) {
    let errors ..= [ "TOOLTIPFORMATTER parameter must be a string" ];
    let tooltipFormatter = nil;
}

// debug
var debug = $6 ?? $debug ?? false;

// == Convenience Parameters ==

// animation
var animation = $animation ?? false;
if (animation is not bool) {
    let errors ..= [ "ANIMATION parameter must be bool" ];
    let animation = false;
}

// categories
var categories = $categories;
if (categories is not nil && categories is not list) {
    let errors ..= [ "CATEGORIES must be a list of category names" ];
    let categories = nil;
}

// stacking
var stacking = $stacking;
if (stacking is not nil && !list.contains(["normal", "percent"], stacking
    let errors ..= [ "STACKING parameter must be either 'normal' or 'perc
    let stacking = nil;
}

// title
var title = $title;
if (title is not nil && title is not str) {
    let errors ..= [ "TITLE parameter must be a string" ];
    let title = nil;
}

// Construct the chart options
```

```
var chartOptions;
if (type == raw) {
    let chartOptions = { chart: { renderTo: @holder }, series: data };
}
else {
    // Chart: should be derived from user-specified options
    var titleMargin  = (#title ? 40 : 0);
    var topMargin    = titleMargin + (type == pie ? 0 : (titleMargin ? 40
    var bottomMargin = (type == pie ? 0 : 30);
    var leftMargin   = (type == pie ? 30 : 60);
    var rightMargin  = (type == pie ? 0 : 10);
    var specChart = {
        renderTo: @holder,
        defaultSeriesType: type,
        margin: [ topMargin, rightMargin, bottomMargin, leftMargin ],
        borderRadius: 5
    };

    // Tooltip
    var specTooltip = {};

    // plotOptions: Very specific to chart type
    var specPlotOptions = { series: { animation: animation, stacking: sta
    if (type == pie) let specPlotOptions ..= {
        pie: {
            allowPointSelect: true,
            dataLabels: {
                enabled: true,
                color: "white"
            },
            center: [ num.min(size[0],size[1]) / 2 + 10, "50%" ]
        }
    };

    // Credits: turn off "Highcharts" credit
    var specCredits = { enabled: false };

    // Title
    var specTitle = {
        text:title,
        x: 25,
        y: 30,
        style: {
            color: "black",
            fontWeight: "bold",
```

```
        fontSize: "18px"
    }
};

// Legend
var specLegend =
    type == pie ?
    {
        layout: "vertical",
        borderRadius: 5,
        align: "right",
        verticalAlign: "middle",
        x: -30,
        y: 0
    } :
    {
        layout: "horizontal",
        borderRadius: 5,
        verticalAlign: "top",
        x: 25,
        y: (#title ? 45 : 10)
    };

// X axis
var specXaxis = {
    categories : categories
};

// Y axis
var specYaxis = {
};

// Series
var specSeries =
    type == pie ?
    [ { data: data } ] :
    data;

// Assemble it all
let chartOptions = {
    chart: specChart,
    tooltip: specTooltip,
    plotOptions: specPlotOptions,
    credits: specCredits,
    title: specTitle,
```

```
        legend: specLegend,
        xAxis: specXaxis,
        yAxis: specYaxis,
        series: specSeries
    };
}

// Merge user-supplied options into chartOptions
foreach (var k:v in options) {
    if (v is not map) let errors ..= [ "OPTIONS arg: '"..k.."' element mu
    let chartOptions ..= {
        (k) : (chartOptions[k]??{}) .. { (kk):(vv is map ? chartOptions[k
    };
}
```

Output Generation

```
if (#errors) <div style="color:red">
    <strong> "Highcharts errors:" </strong>;
    <ul> foreach (var e in errors) <li> e </li>; </ul>;
</div>;
else <html>
<head>

<script type="text/javascript" src=(thisTemplate.files["highcharts.js"].u
<script type="text/javascript">
" $(function() {
    var options = "..json.emit(chartOptions)..";
";
// Add formatter(s) if necessary
if (labelFormatter is not nil) "
    if (options.plotOptions.series.dataLabels == null) options.plotOption
    options.plotOptions.series.dataLabels.formatter = function(){ "..labe
";

if (tooltipFormatter is not nil) "
    if (options.tooltip == null) options.tooltip = {};
    options.tooltip.formatter = function(){ "..tooltipFormatter.." };
";

// Resume
"   var chart = new Highcharts.Chart(options);
});
" </script>;
```

```
</head>;
<body>

if (debug) <div style="color:red; border: 1px dotted red; padding:10px;">
    <strong> "Highcharts debug output" </strong>;
  <div>
    "Here are the parameters passed to the ";
    web.link("http://developer.mindtouch.com/User:neilw/Templates_and_Ext
             "template"); ":";
    <div style="margin-left: 20px"> prettyPrint(args) </div>;
    "Here is the generated argument map passed to "; web.link("http://www
    <div style="margin-left: 20px"> prettyPrint(chartOptions) </div>;
    if (labelFormatter is not nil)
        <div> "The custom datalabel formatter is: "; <strong> "function()
    if (tooltipFormatter is not nil)
        <div> "The custom tooltip formatter is: "; <strong> "function() {
    <div> "The resulting chart appears below this box." </div>;
  </div>;
</div>;

<div id=(@holder) style=("width:"..size[0].."px; height:"..size[1].."px;"

</body>;
</html>;
```