MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart Functional Specification

NetApp Confidential

< [MEG](https://wikid.netapp.com/w/MEG)‎ | [UXE](https://wikid.netapp.com/w/MEG/UXE)‎ | [NWF](https://wikid.netapp.com/w/MEG/UXE/NWF)‎ | [Releases](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases)‎ | [Bananaquit](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit)‎ | [Specifications](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications)

|  |  |  |  |
| --- | --- | --- | --- |
| ***Version 1.0*** | ***Oct 16 2009*** | ***Approved*** | ***dl-nwf-architecture*** |

**Contents**

 [[hide](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification)]

* [1 Overview](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Overview)
* [2 Requirements](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Requirements)
* [3 Related Documents](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Related_Documents)
* [4 Terminology](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Terminology)
* [5 Dependencies](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Dependencies)
* [6 Detailed Use Cases and Flow Diagrams](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Detailed_Use_Cases_and_Flow_Diagrams)
* [7 User Interfaces](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#User_Interfaces)
  + [7.1 Command Line Interface](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Command_Line_Interface)
  + [7.2 Graphical Interface](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Graphical_Interface)
    - [7.2.1 FusionChart in Dashboard](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#FusionChart_in_Dashboard)
    - [7.2.2 Printing a FusionChart](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Printing_a_FusionChart)
* [8 Programmatic Interfaces](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Programmatic_Interfaces)
  + [8.1 Client-Server APIs](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Client-Server_APIs)
  + [8.2 Client-Only APIs](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Client-Only_APIs)
    - [8.2.1 Dashboard](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Dashboard)
    - [8.2.2 Chart](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Chart)
    - [8.2.3 ChartFactory](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartFactory)
    - [8.2.4 ChartType](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartType)
    - [8.2.5 FusionChartType](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#FusionChartType)
    - [8.2.6 PowerChartType](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#PowerChartType)
    - [8.2.7 ChartEventHandler](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartEventHandler)
    - [8.2.8 ChartEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartEvent)
    - [8.2.9 ChartLoadedEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartLoadedEvent)
    - [8.2.10 ChartRenderedEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartRenderedEvent)
    - [8.2.11 ChartDataLoadedEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartDataLoadedEvent)
    - [8.2.12 ChartDataLoadErrorEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartDataLoadErrorEvent)
    - [8.2.13 ChartNoDataToDisplayEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartNoDataToDisplayEvent)
    - [8.2.14 ChartDataXmlInvalidEvent](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartDataXmlInvalidEvent)
    - [8.2.15 FusionChart](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#FusionChart)
    - [8.2.16 FusionChartJavaScriptObject](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#FusionChartJavaScriptObject)
    - [8.2.17 InvalidChartStateException](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#InvalidChartStateException)
    - [8.2.18 ChartNotRenderedException](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#ChartNotRenderedException)
  + [8.3 Usage](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Usage)
    - [8.3.1 Setting chart data via setDataXml()](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Setting_chart_data_via_setDataXml.28.29)
      * [8.3.1.1 EventService interaction](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#EventService_interaction)
    - [8.3.2 Setting chart data via setDataUrl()](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Setting_chart_data_via_setDataUrl.28.29)
    - [8.3.3 Handling events generated by a Chart](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Handling_events_generated_by_a_Chart)
* [9 RAS (Reliability, Availability, Serviceability)](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#RAS_.28Reliability.2C_Availability.2C_Serviceability.29)
  + [9.1 Error reporting](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Error_reporting)
* [10 Test Strategies](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Test_Strategies)
* [11 Revision History](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Revision_History)
* [12 Approvals](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Approvals)

Overview[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=1)]

This document describes an API that allows applications to include FusionCharts Flash-based charts. It provides a factory that can create a chart wrapped in a GWT HTML Widget, so that it can be added anywhere a GWT Widget can (most likely in a Dashboard).

Requirements[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=2)]

* NWF must provide a Java API that allows applications to include FusionCharts.
* NWF must not require application developers to write JSNI code (to call the FusionCharts javascript API directly) in order to create and/or modify FusionCharts.
* NWF must provide a way for application developers to add FusionCharts created by the Java API to an NWF Dashboard.

Related Documents[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=3)]

* [FusionCharts V3 Documentation](http://www.fusioncharts.com/docs/)

Terminology[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=4)]

* **FusionChart** - a flash-based chart generated by FusionCharts V3 from InfoSoft Global

Dependencies[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=5)]

Detailed Use Cases and Flow Diagrams[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=6)]

* An application developer using NWF needs to include a FusionCharts-based chart to an NWF Dashboard in their application.

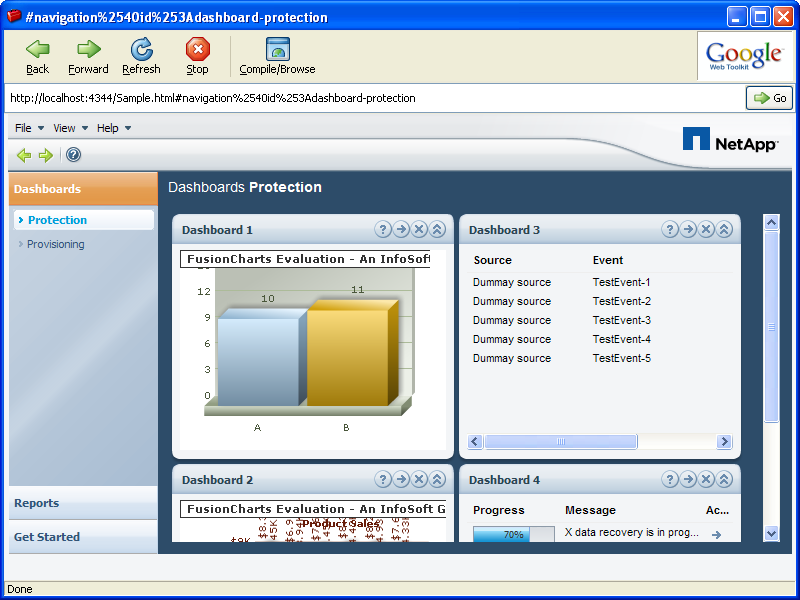
User Interfaces[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=7)]

Command Line Interface[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=8)]

Graphical Interface[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=9)]

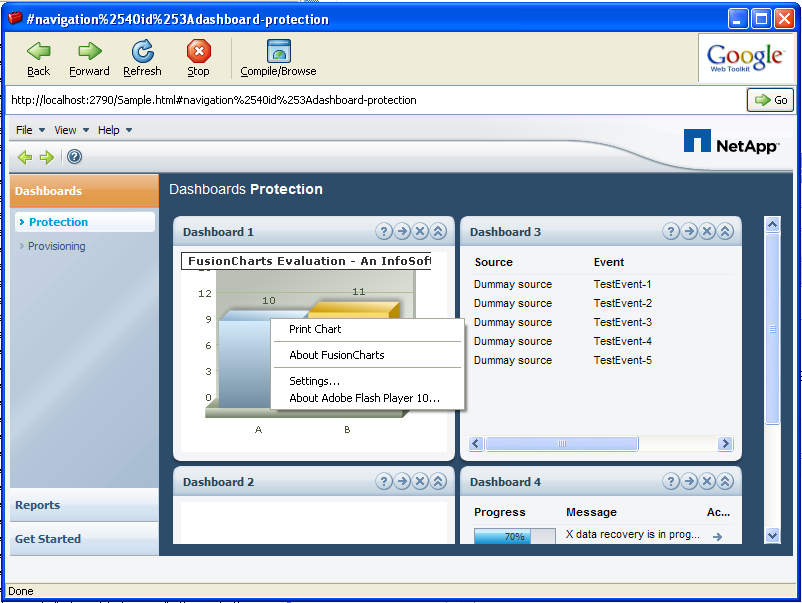
**FusionChart in Dashboard**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=10)]

In the screenshot below, Dashboard 1 contains a FusionChart that was created and included using the code from the [Usage](https://wikid.netapp.com/w/MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification#Usage) section below.

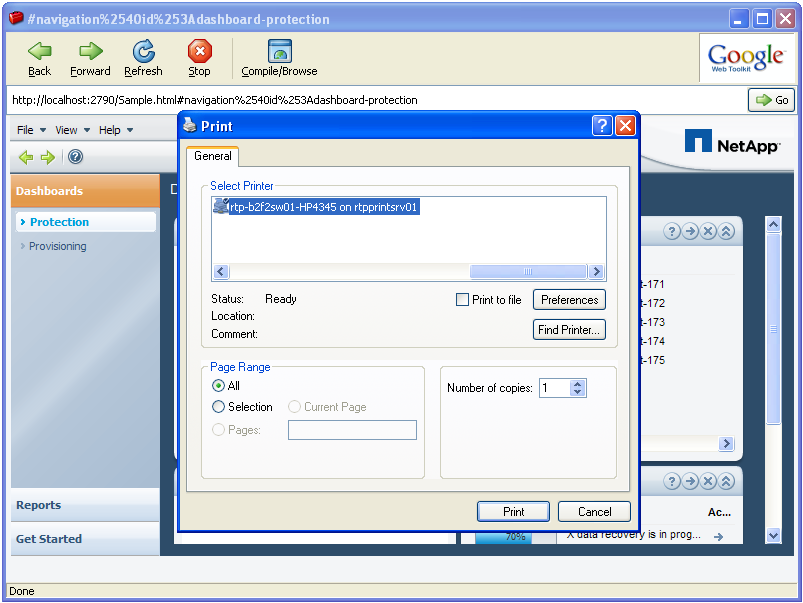
[](https://wikid.netapp.com/w/File:Fusionchart_nwf_dashboard.PNG)

**Printing a FusionChart**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=11)]

The screenshot below shows the "Print Chart" option that's available in the right-click menu of the FusionChart.

[](https://wikid.netapp.com/w/File:Fusionchart_print_menu.png)

If the "Print Chart" option is selected, the standard print dialog will be displayed, as in the screenshot below.

[](https://wikid.netapp.com/w/File:Fusionchart_print_dialog.png)

Programmatic Interfaces[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=12)]

Client-Server APIs[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=13)]

Client-Only APIs[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=14)]

**Dashboard**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=15)]

Dashboard's constructor will need to be updated to take a GWT Widget as a parameter instead of a GXT Component. This won't break any existing code, since Component extends Widget.

**public** Dashboard(Widget content);

**public** Dashboard(Widget content, String id);

**Chart**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=16)]

Chart is an abstraction of the various different kinds of charts we support. This will allow us to add charts from other vendors in the future without breaking interfaces.

***/\*\****

***\* An HTML widget that contains a rendered chart.***

***\*/***

**public** **abstract** **class** Chart **extends** HTML {

***/\*\****

***\* A Map that maps the id of every chart that has been created to the***

***\* HandlerManager associated with that chart, so that static methods***

***\* can have access to the HandlerManager associated with a chart.***

***\*/***

**protected** **static** Map<String, HandlerManager> handlerManagerByChartId = **new** HashMap<String, HandlerManager>();

***/\*\****

***\* Constructs a chart object using the given html string.***

***\****

***\* @param id the id of the chart***

***\* @param html the html string to use to show the chart***

***\*/***

**protected** Chart(String id, String html);

***/\*\****

***\* Returns the id of the chart. This id is unique among all charts in the***

***\* application.***

***\****

***\* @return the id of the chart***

***\*/***

**public** String getId();

***/\*\****

***\* Returns the xml data of the chart. This method can be used to retrieve***

***\* the data rendered by the chart regardless of whether this data was set***

***\* using setDataXml() or setDataUrl().***

***\****

***\* @return the xml data of the chart***

***\*/***

**public** **abstract** String getDataXml();

***/\*\****

***\* Sets the xml that is used to provide data to be rendered by the chart.***

***\* This method can be used to dynamically update the chart once it's been***

***\* rendered.***

***\****

***\* @param xml the xml with the data to be rendered by the graph.***

***\*/***

**public** **abstract** **void** setDataXml(String xml);

***/\*\****

***\* Returns the url that the chart queries to get its XML data.***

***\****

***\* @return the url that the chart queries to get its XML data***

***\*/***

**public** **abstract** String getDataUrl();

***/\*\****

***\* Specifies a url to the xml data to be rendered by the chart. This***

***\* method can be used to dynamically update the chart once it's been***

***\* rendered.***

***\****

***\* @param url a url to the xml data to be rendered by the chart***

***\*/***

**public** **abstract** **void** setDataUrl(String url);

***/\*\****

***\* Gets the value of the attribute with the specified name.***

***\****

***\* @param attrName the name of the attribute***

***\* @return the value of the attribute***

***\*/***

**public** **abstract** String getAttribute(String attrName);

***/\*\****

***\* Sets the value of the specified attribute.***

***\****

***\* @param attrName the name of the attribute***

***\* @param attrValue the value of the attribute***

***\*/***

**public** **abstract** **void** setAttribute(String attrName, String attrValue);

***/\*\****

***\* Indicates whether the chart has been rendered.***

***\****

***\* @return a boolean indicating whether the chart has been rendered***

***\*/***

**public** **abstract** **boolean** hasRendered();

***/\*\****

***\* Initiates the printing of the chart. In most browsers, this will cause***

***\* the browser's print dialog to appear in order for the user to specify***

***\* details for how to print the chart.***

***\*/***

**public** **abstract** **void** print();

***/\*\****

***\* Returns the data of the chart as a CSV string.***

***\****

***\* @return the data of the chart as a CSV string***

***\*/***

**public** **abstract** String getDataAsCsv();

***/\*\****

***\* Returns whether the chart is in transparent mode.***

***\****

***\* @return whether the chart is in transparent mode***

***\*/***

**public** **abstract** **boolean** isTransparent();

***/\*\****

***\* Sets the chart to use transparent mode when the transparent parameter is***

***\* true. If this method is not called, the default is for the chart to be***

***\* transparent.***

***\****

***\* @param transparent should the chart use transparent mode***

***\*/***

**public** **abstract** **void** setTransparent(**boolean** transparent);

***/\*\****

***\* Adds a handler that is notified when specified event occurs. This***

***\* method returns a HandlerRegistration that can be used to remove the***

***\* handler.***

***\****

***\* @param handler the handler to notify when the event occurs***

***\* @return the handler registration that can be used to remove the handler***

***\*/***

**public** <H **extends** ChartEventHandler> HandlerRegistration addChartHandler(GwtEvent.Type<H> type, H handler);

***/\*\****

***\* Sends an event to the handlers.***

***\****

***\* @param event the event to send to the handlers.***

***\*/***

**protected** **static** **void** fireChartEvent(ChartEvent event);

}

**ChartFactory**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=17)]

ChartFactory is used to create Charts based on the ChartType specified. This provides a generic way to construct FusionCharts, PowerCharts, and any other charts that NWF may support in the future.

***/\*\****

***\* A factory that creates instances of Chart.***

***\*/***

**public** **class** ChartFactory {

***/\*\****

***\* Default constructor is private since this is a static utility class and***

***\* should not be instantiated.***

***\*/***

**private** ChartFactory();

***/\*\****

***\* Creates a Chart with the specified chart type, and***

***\* the default width, height, debug mode, and registerWithJavaScript***

***\* values. The id will be auto-generated.***

***\****

***\* @param chartType the type of chart to be created***

***\* @return an instance of Chart***

***\*/***

**public** **static** Chart createChart(ChartType chartType);

***/\*\****

***\* Creates a Chart with the specified id, chart type, and***

***\* the default width, height, debug mode, and registerWithJavaScript***

***\* values.***

***\****

***\* @param chartId the id that should be assigned to the Chart***

***\* @param chartType the type of chart to be created***

***\* @return an instance of Chart***

***\*/***

**public** **static** Chart createChart(String chartId, ChartType chartType);

***/\*\****

***\* Creates a Chart with the specified chart type, width, and height,***

***\* and the default debug mode and registerWithJavaScript values.***

***\* The id will be auto-generated.***

***\****

***\* @param chartType the type of chart to be created***

***\* @param width the width of the chart***

***\* @param height the height of the chart***

***\* @return an instance of Chart***

***\*/***

**public** **static** Chart createChart(ChartType chartType, **int** width, **int** height);

***/\*\****

***\* Creates a Chart with the specified id, chart type, width, and height,***

***\* and the default debug mode and registerWithJavaScript values.***

***\****

***\* @param chartId the id that should be assigned to the Chart***

***\* @param chartType the type of chart to be created***

***\* @param width the width of the chart***

***\* @param height the height of the chart***

***\* @return an instance of Chart***

***\*/***

**public** **static** Chart createChart(String chartId, ChartType chartType, **int** width, **int** height);

***/\*\****

***\* Creates a Chart with the specified id, chart type, width, height,***

***\* debug mode, and registerWithJavaScript values. The id will be***

***\* auto-generated.***

***\****

***\* @param chartType the type of chart to be created***

***\* @param width the width of the chart***

***\* @param height the height of the chart***

***\* @param debugMode if true, enables a debug window that provide debug info***

***\* @return an instance of Chart***

***\*/***

**public** **static** Chart createChart(ChartType chartType, Integer width, Integer height, **boolean** debugMode);

***/\*\****

***\* Creates a Chart with the specified id, chart type, width, height,***

***\* debug mode, and registerWithJavaScript values.***

***\****

***\* @param chartId the id that should be assigned to the Chart***

***\* @param chartType the type of chart to be created***

***\* @param width the width of the chart***

***\* @param height the height of the chart***

***\* @param debugMode if true, enables a debug window that provide debug info***

***\* @return an instance of Chart***

***\*/***

**public** **static** Chart createChart(String chartId, ChartType chartType, Integer width, Integer height, **boolean** debugMode);

**ChartType**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=18)]

***/\*\****

***\* An interface that the various ChartType enums will implement. Enums of***

***\* this type will know the URL of chart type file.***

***\*/***

**public** **interface** ChartType {

***/\*\****

***\* Returns the absolute URL of the chart type file.***

***\****

***\* @return the absolute URL of the chart type file***

***\*/***

String getChartTypeUrl();

}

**FusionChartType**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=19)]

***/\*\****

***\* An enumeration that lists all available FusionChart types.***

***\*/***

**public** **enum** FusionChartType **implements** ChartType {

Area2D,

Bar2D,

Bubble,

Column2D,

Column3D,

Doughnut2D,

Doughnut3D,

Line,

MSArea,

MSBar2D,

MSBar3D,

MSColumn2D,

MSColumn3D,

MSColumn3DLineDY,

MSColumnLine3D,

MSCombi2D,

MSCombi3D,

MSCombiDY2D,

MSLine,

MSStackedColumn2D,

MSStackedColumn2DLineDY,

Pie2D,

Pie3D,

Scatter,

ScrollArea2D,

ScrollColumn2D,

ScrollCombi2D,

ScrollCombiDY2D,

ScrollLine2D,

ScrollStackedColumn2D,

SSGrid,

StackedArea2D,

StackedBar2D,

StackedBar3D,

StackedColumn2D,

StackedColumn3D,

StackedColumn3DLineDY;

***/\*\****

***\* Returns the absolute URL of the swf file for the chart type.***

***\****

***\* @return the absolute URL of the swf file for the chart type***

***\* @see com.netapp.nwf.client.userinterface.chart.ChartType#getChartTypeUrl()***

***\*/***

@Override

**public** String getChartTypeUrl() {

**return** GWT.getModuleBaseURL() + "charts/fusioncharts/" + **this**.toString() + ".swf";

}

}

**PowerChartType**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=20)]

***/\*\****

***\* An enumeration that lists all available PowerChart types.***

***\*/***

**public** **enum** PowerChartType **implements** ChartType {

CandleStick,

DragArea,

DragColumn2D,

DragLine,

DragNode,

ErrorBar2D,

InverseMSArea,

InverseMSColumn2D,

InverseMSLine,

Kagi,

LogMSColumn2D,

LogMSLine,

MSSpline,

MSSplineArea,

MultiAxisLine,

MultiLevelPie,

Radar,

SelectScatter,

Spline,

SplineArea,

Waterfall2D;

***/\*\****

***\* Returns the absolute URL of the swf file for the chart type.***

***\****

***\* @return the absolute URL of the swf file for the chart type***

***\* @see com.netapp.nwf.client.userinterface.chart.ChartType#getChartUrl()***

***\*/***

@Override

**public** String getChartUrl() {

**return** GWT.getModuleBaseURL() + "charts/powercharts/" + **this**.toString() + ".swf";

}

}

**ChartEventHandler**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=21)]

***/\*\****

***\* An event handler that handles events produced by charts.***

***\*/***

**public** **interface** ChartEventHandler **extends** EventHandler {

}

**ChartEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=22)]

***/\*\****

***\* An event generated by a chart.***

***\*/***

**public** **abstract** **class** ChartEvent<H **extends** ChartEventHandler, T **extends** GwtEvent.Type<H>> **extends** Event<H, T> {

***/\*\****

***\* The type of the event.***

***\*/***

**public** **static** **class** Type<H> **extends** GwtEvent.Type<H> {

}

***/\*\****

***\* This constructor is for use by subclasses of ChartEvent, since***

***\* ChartEvent is abstract and cannot be instantiated.***

***\****

***\* @param type***

***\*/***

**protected** ChartEvent(T type);

***/\*\****

***\* Returns the id of the chart that produced the event.***

***\****

***\* @return the id of the chart that produced the event***

***\*/***

**public** **abstract** String getChartId();

}

**ChartLoadedEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=23)]

***/\*\****

***\* Event used to notify that a chart has been loaded.***

***\*/***

**public** **class** ChartLoadedEvent **extends** ChartEvent<ChartLoadedEvent.Handler, ChartLoadedEvent.Type> {

***/\*\****

***\* The event handler.***

***\*/***

**public** **interface** Handler **extends** ChartEventHandler {

**void** onChartLoaded(ChartLoadedEvent event);

}

***/\*\****

***\* The event type class.***

***\*/***

**protected** **static** **class** Type **extends** EventType<Handler> {

}

***/\*\****

***\* The id of the chart that was loaded.***

***\*/***

**protected** String chartId;

***/\*\****

***\* The event type instance.***

***\*/***

**public** **static** Type TYPE = **new** Type();

***/\*\****

***\* Constructs a ChartLoadedEvent for the chart whose id is specified.***

***\****

***\* @param chartId the id of the chart that was loaded***

***\*/***

**public** ChartLoadedEvent(String chartId);

***/\*\****

***\* Returns the id of the chart that was loaded.***

***\****

***\* @return the id of the chart that was loaded***

***\*/***

**public** String getChartId();

***/\*\****

***\* Dispatches the event to the handler.***

***\****

***\* @see com.google.gwt.event.shared.GwtEvent#dispatch(com.google.gwt.event.shared.EventHandler)***

***\*/***

@Override

**protected** **void** dispatch(Handler handler);

}

**ChartRenderedEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=24)]

***/\*\****

***\* Event used to notify that a chart has been rendered.***

***\*/***

**public** **class** ChartRenderedEvent **extends** ChartEvent<ChartLoadedEvent.Handler, ChartLoadedEvent.Type> {

***/\*\****

***\* The event handler.***

***\*/***

**public** **interface** Handler **extends** ChartEventHandler {

**void** onChartRendered(ChartRenderedEvent event);

}

***/\*\****

***\* The event type class.***

***\*/***

**protected** **static** **class** Type **extends** EventType<Handler> {

}

***/\*\****

***\* The id of the chart that was rendered.***

***\*/***

**protected** String chartId;

***/\*\****

***\* The event type instance.***

***\*/***

**public** **static** Type TYPE = **new** Type();

***/\*\****

***\* Constructs a ChartRenderedEvent for the chart whose id is specified.***

***\****

***\* @param chartId the id of the chart that was rendered***

***\*/***

**public** ChartRenderedEvent(String chartId);

***/\*\****

***\* Returns the id of the chart that was rendered.***

***\****

***\* @return the id of the chart that was rendered***

***\*/***

**public** String getChartId();

***/\*\****

***\* Dispatches the event to the handler.***

***\****

***\* @see com.google.gwt.event.shared.GwtEvent#dispatch(com.google.gwt.event.shared.EventHandler)***

***\*/***

@Override

**protected** **void** dispatch(Handler handler);

}

**ChartDataLoadedEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=25)]

***/\*\****

***\* Event used to notify that a chart's data has been loaded.***

***\*/***

**public** **class** ChartDataLoadedEvent **extends** ChartEvent<ChartLoadedEvent.Handler, ChartLoadedEvent.Type> {

***/\*\****

***\* The event handler.***

***\*/***

**public** **interface** Handler **extends** ChartEventHandler {

**void** onChartDataLoaded(ChartDataLoadedEvent event);

}

***/\*\****

***\* The event type class.***

***\*/***

**protected** **static** **class** Type **extends** EventType<Handler> {

}

***/\*\****

***\* The id of the chart whose data was loaded.***

***\*/***

**protected** String chartId;

***/\*\****

***\* The event type instance.***

***\*/***

**public** **static** Type TYPE = **new** Type();

***/\*\****

***\* Constructs a ChartDataLoadedEvent for the chart whose id is specified.***

***\****

***\* @param chartId the id of the chart whose data has been loaded***

***\*/***

**public** ChartDataLoadedEvent(String chartId);

***/\*\****

***\* Returns the id of the chart whose data has been loaded.***

***\****

***\* @return the id of the chart whose data has been loaded***

***\*/***

**public** String getChartId();

***/\*\****

***\* Dispatches the event to the handler.***

***\****

***\* @see com.google.gwt.event.shared.GwtEvent#dispatch(com.google.gwt.event.shared.EventHandler)***

***\*/***

@Override

**protected** **void** dispatch(Handler handler);

}

**ChartDataLoadErrorEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=26)]

***/\*\****

***\* Event used to notify that an error occurred while loading the data for a***

***\* chart.***

***\*/***

**public** **class** ChartDataLoadErrorEvent **extends** ChartEvent<ChartLoadedEvent.Handler, ChartLoadedEvent.Type> {

***/\*\****

***\* The event handler.***

***\*/***

**public** **interface** Handler **extends** ChartEventHandler {

**void** onChartDataLoadError(ChartDataLoadErrorEvent event);

}

***/\*\****

***\* The event type class.***

***\*/***

**protected** **static** **class** Type **extends** EventType<Handler> {

}

***/\*\****

***\* The id of the chart that caused the error.***

***\*/***

**protected** String chartId;

***/\*\****

***\* The event type instance.***

***\*/***

**public** **static** Type TYPE = **new** Type();

***/\*\****

***\* Constructs a ChartDataLoadErrorEvent for the chart whose id is specified.***

***\****

***\* @param chartId the id of the chart that caused the error***

***\*/***

**public** ChartDataLoadErrorEvent(String chartId);

***/\*\****

***\* Returns the id of the chart that caused the error.***

***\****

***\* @return the id of the chart that caused the error***

***\*/***

**public** String getChartId();

***/\*\****

***\* Dispatches the event to the handler.***

***\****

***\* @see com.google.gwt.event.shared.GwtEvent#dispatch(com.google.gwt.event.shared.EventHandler)***

***\*/***

@Override

**protected** **void** dispatch(Handler handler);

}

**ChartNoDataToDisplayEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=27)]

***/\*\****

***\* Event used to notify that a chart has no data to display.***

***\*/***

**public** **class** ChartNoDataToDisplayEvent **extends** ChartEvent<ChartLoadedEvent.Handler, ChartLoadedEvent.Type> {

***/\*\****

***\* The event handler.***

***\*/***

**public** **interface** Handler **extends** ChartEventHandler {

**void** onChartNoDataToDisplay(ChartNoDataToDisplayEvent event);

}

***/\*\****

***\* The event type class.***

***\*/***

**protected** **static** **class** Type **extends** EventType<Handler> {

}

***/\*\****

***\* The id of the chart that has no data to display.***

***\*/***

**protected** String chartId;

***/\*\****

***\* The event type instance.***

***\*/***

**public** **static** Type TYPE = **new** Type();

***/\*\****

***\* Constructs a ChartNoDataToDisplayEvent for the chart whose id is***

***\* specified.***

***\****

***\* @param chartId the id of the chart that has no data to display***

***\*/***

**public** ChartNoDataToDisplayEvent(String chartId);

***/\*\****

***\* Returns the id of the chart that has no data to display.***

***\****

***\* @return the id of the chart that has no data to display***

***\*/***

**public** String getChartId();

***/\*\****

***\* Dispatches the event to the handler.***

***\****

***\* @see com.google.gwt.event.shared.GwtEvent#dispatch(com.google.gwt.event.shared.EventHandler)***

***\*/***

@Override

**protected** **void** dispatch(Handler handler);

}

**ChartDataXmlInvalidEvent**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=28)]

***/\*\****

***\* Event used to notify that a chart's data xml is invalid.***

***\*/***

**public** **class** ChartDataXmlInvalidEvent **extends** ChartEvent<ChartLoadedEvent.Handler, ChartLoadedEvent.Type> {

***/\*\****

***\* The event handler.***

***\*/***

**public** **interface** Handler **extends** ChartEventHandler {

**void** onChartDataXmlInvalid(ChartDataXmlInvalidEvent event);

}

***/\*\****

***\* The event type class.***

***\*/***

**protected** **static** **class** Type **extends** EventType<Handler> {

}

***/\*\****

***\* The id of the chart whose data xml is invalid.***

***\*/***

**protected** String chartId;

***/\*\****

***\* The event type instance.***

***\*/***

**public** **static** Type TYPE = **new** Type();

***/\*\****

***\* Constructs a ChartDataXmlInvalidEvent for the chart whose id is specified.***

***\****

***\* @param chartId the id of the chart whose data xml is invalid.***

***\*/***

**public** ChartDataXmlInvalidEvent(String chartId);

***/\*\****

***\* Returns the id of the chart whose xml data is invalid.***

***\****

***\* @return the id of the chart whose xml data is invalid***

***\*/***

**public** String getChartId();

***/\*\****

***\* Dispatches the event to the handler.***

***\****

***\* @see com.google.gwt.event.shared.GwtEvent#dispatch(com.google.gwt.event.shared.EventHandler)***

***\*/***

@Override

**protected** **void** dispatch(Handler handler);

}

**FusionChart**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=29)]

FusionChart wraps a FusionChart returned by the FusionChart javascript API in a GWT HTML widget.

***/\*\****

***\* An HTML widget that contains a rendered FusionChart.***

***\*/***

**public** **class** FusionChart **extends** Chart {

***/\*\****

***\* Constructs a FusionChart object from a FusionChartJavaScriptObject.***

***\****

***\* @param javaScriptObject the FusionChartJavaScriptObject from which to***

***\* create the FusionChart***

***\*/***

**public** FusionChart(**final** FusionChartJavaScriptObject javaScriptObject);

***/\*\****

***\* This method is called by GWT when the FusionChart widget has been added***

***\* to the browser's document. The chart is actually rendered in this***

***\* method since the widget must be added to the browser's document before***

***\* any rendering can be done.***

***\* <br />***

***\* <b>Note:</b> This method should not be called directly by developers.***

***\****

***\* @see com.google.gwt.user.client.ui.Widget#onAttach()***

***\*/***

@Override

**public** **void** onAttach();

***/\*\****

***\* Returns the id of the FusionChartJavaScriptObject. This id should be***

***\* unique among all charts in the application.***

***\****

***\* @return the id of the FusionChartJavaScriptObject***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#getId()***

***\*/***

@Override

**public** String getId();

***/\*\****

***\* Returns the xml data of the chart. This method can be used to retrieve***

***\* the data rendered by the chart regardless of whether this data was set***

***\* using setDataXml() or setDataUrl().***

***\****

***\* @return the XML data of the chart***

***\* @throws ChartNotRenderedException if called before the chart is rendered***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#getDataXml()***

***\*/***

@Override

**public** String getDataXml();

***/\*\****

***\* Sets the xml that's used to provide data to be rendered by the chart.***

***\* This method can be used to dynamically update the chart once it's been***

***\* rendered.***

***\* <br />***

***\* The format of this xml is described in the FusionCharts v3***

***\* Documentation, which can be found here:***

***\* http://www.fusioncharts.com/docs/.***

***\****

***\* @param xml the xml with the data to be rendered by the graph.***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#setDataXml(String)***

***\*/***

@Override

**public** **void** setDataXml(String xml);

***/\*\****

***\* Returns the url that the chart queries to get its XML data.***

***\****

***\* @return the url that the chart queries to get its XML data***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#getDataUrl()***

***\*/***

@Override

**public** String getDataUrl();

***/\*\****

***\* Specifies a url to the xml data to be rendered by the chart. This***

***\* method can be used to dynamically update the chart once it's been***

***\* rendered.***

***\****

***\* @param url a url to the xml data to be rendered by the chart***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#setDataUrl(String)***

***\*/***

@Override

**public** **void** setDataUrl(String url);

***/\*\****

***\* Gets the value of the attribute with the specified name.***

***\* <br />***

***\* Descriptions of the available attributes can be found in the***

***\* "Quick Guide to Chart XML" in the FusionCharts V3 Documentation at:***

***\* http://www.fusioncharts.com/docs/.***

***\****

***\* @param attrName the name of the attribute***

***\* @return the value of the attribute***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#getAttribute(String)***

***\*/***

@Override

**public** String getAttribute(String attrName);

***/\*\****

***\* Sets the value of the specified attribute.***

***\* <br />***

***\* Descriptions of the available attributes can be found in the "Quick***

***\* Guide to Chart XML" in the FusionCharts V3 Documentation at:***

***\* http://www.fusioncharts.com/docs/.***

***\****

***\* @param attrName the name of the attribute***

***\* @param attrValue the value of the attribute***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#setAttribute(String, String)***

***\*/***

@Override

**public** **void** setAttribute(String attrName, String attrValue);

***/\*\****

***\* Indicates whether the chart has been rendered.***

***\****

***\* @return a boolean indicating whether the chart has been rendered***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#hasRendered()***

***\*/***

@Override

**public** **boolean** hasRendered();

***/\*\****

***\* Initiates the printing of the chart. In most browsers, this will cause***

***\* the browser's print dialog to appear in order for the user to specify***

***\* details for how to print the chart.***

***\****

***\* @throws ChartNotRenderedException if called before the chart is rendered***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#print()***

***\*/***

@Override

**public** **void** print();

***/\*\****

***\* Returns the data of the chart as a CSV string.***

***\****

***\* @return the data of the chart as a CSV string***

***\* @throws ChartNotRenderedException if called before the chart is rendered***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#getDataAsCsv()***

***\*/***

@Override

**public** String getDataAsCsv();

***/\*\****

***\* Returns whether the chart is in transparent mode.***

***\****

***\* @return whether the chart is in transparent mode***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#isTransparent()***

***\*/***

**public** **boolean** isTransparent();

***/\*\****

***\* Sets the chart to use transparent mode when the transparent parameter is***

***\* true. If this method is not called, the default is for the chart to be***

***\* transparent.***

***\****

***\* @param transparent should the chart use transparent mode***

***\* @see com.netapp.nwf.client.userinterface.chart.Chart#setTransparent(boolean)***

***\*/***

**public** **void** setTransparent(**boolean** transparent);

}

**FusionChartJavaScriptObject**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=30)]

***/\*\****

***\* A JSNI wrapper around the Javascript FusionCharts variable from***

***\* FusionCharts.js.***

***\*/***

**public** **class** FusionChartJavaScriptObject **extends** JavaScriptObject {

***/\*\****

***\* The default constructor is protected since NWF code should use***

***\* ChartFactory to instantiate a FusionChartJavaScriptObject (and***

***\* encapsulate it in a component).***

***\*/***

**protected** FusionChartJavaScriptObject();

***/\*\****

***\* Returns the id of the FusionChartJavaScriptObject. This id should be***

***\* unique among all charts in the application.***

***\****

***\* @return the id of the FusionChartJavaScriptObject***

***\*/***

**public** **final** **native** String getId();

***/\*\****

***\* Renders the chart in the specified div.***

***\****

***\* @param divId the id of the div where the chart should be rendered***

***\*/***

**public** **final** **native** **void** render(String divId);

***/\*\****

***\* Returns the XML data of the chart.***

***\****

***\* @return the XML data of the chart***

***\*/***

**public** **final** **native** String getXml();

***/\*\****

***\* Sets the xml that's used to provide data to be rendered by the chart.***

***\* This method can be used to dynamically update the chart once it's been***

***\* rendered.***

***\* <br />***

***\* The format of this xml is described in the FusionCharts v3***

***\* Documentation, which can be found here:***

***\* http://www.fusioncharts.com/docs/.***

***\****

***\* @param xml the xml with the data to be rendered by the graph.***

***\*/***

**public** **final** **native** **void** setDataXml(String xml);

***/\*\****

***\* Specifies a url to the xml data to be rendered by the chart. This***

***\* method can be used to dynamically update the chart once it's been***

***\* rendered.***

***\****

***\* @param url a url to the xml data to be rendered by the chart***

***\*/***

**public** **final** **native** **void** setDataUrl(String url);

***/\*\****

***\* Gets the value of the attribute with the specified name.***

***\****

***\* @param attrName the name of the attribute***

***\* @return the value of the attribute***

***\*/***

**public** **final** **native** String getAttribute(String attrName);

***/\*\****

***\* Sets the value of the specified attribute.***

***\****

***\* @param attrName the name of the attribute***

***\* @param attrValue the value of the attribute***

***\*/***

**public** **final** **native** **void** setAttribute(String attrName, String attrValue);

***/\*\****

***\* Indicates whether the chart has been rendered.***

***\****

***\* @return a boolean indicating whether the chart has been rendered***

***\*/***

**public** **final** **native** **boolean** hasRendered();

***/\*\****

***\* Initiates the printing of the chart. In most browsers, this will cause***

***\* the browser's print dialog to appear in order for the user to specify***

***\* details for how to print the chart.***

***\*/***

**public** **final** **native** **void** print();

***/\*\****

***\* Returns the data of the chart as a CSV string.***

***\****

***\* @return the data of the chart as a CSV string***

***\*/***

**public** **final** **native** String getDataAsCsv();

***/\*\****

***\* Sets the chart to use transparent mode when the transparent parameter is***

***\* true. If this method is not called, the default is for the chart to be***

***\* transparent.***

***\****

***\* @param transparent should the chart use transparent mode***

***\*/***

**public** **final** **native** **void** setTransparent(**boolean** transparent);

}

**InvalidChartStateException**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=31)]

InvalidChartStateException is used to indicate that an action was attempted against a chart when the chart was not in a valid state for that action.

***/\*\****

***\* An exception that indicates that a chart is in an invalid state for the***

***\* action that was attempted.***

***\*/***

**public** **class** InvalidChartStateException **extends** Exception {

***/\*\****

***\* Constructor that includes a message to further explain the exception.***

***\****

***\* @param message a message to further explain the exception***

***\*/***

**public** InvalidChartStateException(String message);

}

**ChartNotRenderedException**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=32)]

ChartNotRenderedException is a special case of InvalidChartStateException representing the case where actions are attempted against a chart before it is rendered.

***/\*\****

***\* An exception that indicates that a chart has not been rendered yet.***

***\*/***

**public** **class** ChartNotRenderedException **extends** InvalidChartStateException {

***/\*\****

***\* Default constructor that includes an informational message warning that***

***\* the chart has not yet been rendered and that this is necessary before***

***\* attempting the requested action.***

***\*/***

**public** ChartNotRenderedException();

}

Usage[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=33)]

**Setting chart data via setDataXml()**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=34)]

An application developer could use the following code to add a FusionChart to a dashboard in NWF and set its data using a programmatically defined XML string:

Chart chart = ChartFactory.createChart(FusionChartType.Column3D);

chart.setDataXml("<chart><set label='A' value='" + aValue

+ "' /><set label='B' value='" + bValue + "' /></chart>");

Dashboard db1 = **new** Dashboard(chart, "dashboard1");

**EventService interaction**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=35)]

The NWF Event Service can be used to allow the server to push new data to a Chart. In order to allow this, a few additions will be needed:

* The server would need to be updated to send the updated data xml in an event. An example of how the json sent from the server could look follows:

{

"eventType":"chartDataChanged",

"eventId": <id>,

"response": {

"chartId": "<chart id>",

"chartDataXml":"<xml data>"

}

}

* A javascript overlay would need to be created for the response object defined in the previous step. If the previous example is used, the corresponding javascript overlay would be:

**public** **class** ChartDataChangedEventResponse **extends** JavaScriptObject {

**protected** ChartDataChangedEventResponse() {

}

**public** **final** **native** String getChartId() */\*-{*

*return this.chartId;*

*}-\*/*;

**public** **final** **native** String getDataXml() */\*-{*

*return this.chartDataXml;*

*}-\*/*;

}

}

* A new event type would need to be created to represent that chart data has changed:

**public** **class** ChartDataChangedEventType **extends** EventType {

@Override

**public** String getType() {

**return** "ChartDataChanged";

}

}

* The event service would need to subscribe to the new event:

EventService eventService = EventService.get();

eventService.start(eventServiceUrl);

eventService.subscribe(

**new** EventListener() {

@Override

**public** **void** eventsReceived(List<Event> eventList) {

**for** (Event event : eventList) {

ChartDataChangedEventResponse response = (ChartEventResponse) event.getResponse();

chart.setDataXml(response.getDataXml());

}

}

},

**new** ChartDataChangedEventType());

**Setting chart data via setDataUrl()**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=36)]

An application developer could use the following code to add a FusionChart to a dashboard in NWF and set its data using a URL to an XML file:

Chart chart = ChartFactory.createChart(FusionChartType.Column3D);

chart.setDataUrl(url);

To cause the chart to poll the URL every 30 seconds and update chart accordingly, the following could be added:

*// create a timer that will the re-set the chart data based on the data currently at the URL*

Timer t = **new** Timer() {

@Override

**public** **void** run() {

chart.setDataUrl(url);

}

};

*// schedule the timer to run every 30 seconds.*

t.scheduleRepeating(30000);

**Handling events generated by a Chart**[[edit](https://wikid.netapp.com/wikid/index.php?title=MEG/UXE/NWF/Releases/Bananaquit/Specifications/FusionChart_Functional_Specification&action=edit&section=37)]

The Chart class uses the GWT 1.6 Event Handling pattern. To be notified of an event from the chart, a Handler for that event must be added to the chart. The code below shows how one would add handlers for all the events currently supported by Chart:

chart.addChartHandler(ChartLoadedEvent.TYPE, **new** ChartLoadedEvent.Handler() {

@Override

**public** **void** onChartLoaded(ChartLoadedEvent event) {

Log.getLogger().debug(**this**, "chart loaded: " + event.getChartId());

}

});

chart.addChartHandler(ChartRenderedEvent.TYPE, **new** ChartRenderedEvent.Handler() {

@Override

**public** **void** onChartRendered(ChartRenderedEvent event) {

Log.getLogger().debug(**this**, "chart rendered: " + event.getChartId());

}

});

chart.addChartHandler(ChartDataLoadedEvent.TYPE, **new** ChartDataLoadedEvent.Handler() {

@Override

**public** **void** onChartDataLoaded(ChartDataLoadedEvent event) {

Log.getLogger().debug(**this**, "chart data loaded: " + event.getChartId());

}

});

chart.addChartHandler(ChartDataLoadErrorEvent.TYPE, **new** ChartDataLoadErrorEvent.Handler() {

@Override

**public** **void** onChartDataLoadError(ChartDataLoadErrorEvent event) {

Log.getLogger().error(**this**, "error while loading: " + event.getChartId());

}

});

chart.addChartHandler(ChartNoDataToDisplayEvent.TYPE, **new** ChartNoDataToDisplayEvent.Handler() {

@Override

**public** **void** onChartNoDataToDisplay(ChartNoDataToDisplayEvent event) {

Log.getLogger().warn(**this**, "chart has no data to display: " + event.getChartId());

}

});

chart.addChartHandler(ChartDataXmlInvalidEvent.TYPE, **new** ChartDataXmlInvalidEvent.Handler() {

@Override

**public** **void** onChartDataXmlInvalid(ChartDataXmlInvalidEvent event) {

Log.getLogger().error(**this**, "chart's data xml is invalid: " + event.getChartId());

}

});