..

Custom Style Extensions for Javascript and CSS

Date published: 2020-10-30
Date modified: 2022-09-21

https://docs.cloudera.com/

Legal Notice

Contents

Extension JS API endpoints

JavaScript module returns an object that may implement one of the endpoints: version() on page 4, supported() on page 4, settings() on page 5, disableDraw() on page 6, beforeDraw() on page 6, afterDraw() on page 6, and afterDataRequest() on page 6.

version()

This must exist, and reports on the version of the API that this object implements. Currently supported version strings is "1". This must be a string, not a number.

supported()

If this member exists, it is expected to be an object that contains a member called visualType. This member must point to a string with the type of visual supported, or an array of strings that contain the types of visuals supported.

For example,

```
supported: {
  visualType: "trellis-bars"
},
```

Table 1: Supported Visual Types

Visual Type	Object Name
Table	table
Cross Tabulation	crosstab
Bars	trellis-bars
Lines	trellis-lines
Combo	combo
Areas	trellis-areas
Grouped Bars	trellis-groupedbars
KPI	kpi
Packed Bubbles	packed-bubbles
Scatter	scatter
Flow	flow
Funnel	funnel
Pie	pie
Radial	radial
Chord	chord
Correlation Heatmap	correlation
Correlation Flow	correlation-flow
Calendar Heatmap	calendar-heatmap
Map	map
Interactive Map	leaflet
Sparklines	sparklines

Visual Type	Object Name
External Link	link
Histogram	histogram
Extension	extension
Field Statistics	column-stats
Queries	rawtable
Rich Text	html
Network	network
Dendrogram	dendrogram
Treemap	treemap
Bullet	bullet
Gauge	gauge
Word Cloud	wordcloud
Box Plot	boxplot
Timeline	timeline

settings()

If a member with this name exists, it is expected to be a function that returns an array. Each array element is then expected to be an object with the following members:

```
settings: function() {
  return [
    {
      id: "Column Name"
    },
    {
      id: "Max bar width",
      defaultValue: "50"
    },
    {
      id: "Color",
      defaultValue: "steelblue"
    }
  ];
},
```

Table 2: Settings Specifications

Member	Name	Description	Default Value	Mandatory
id	Column Name	The ID of this setting.	None	Yes
displayName	Column Name	Name used to display this setting.	Same as ID	No
type	String	The data type of the variable. Supported types include String, and bool ean.	String	No
defaultValue	""	The default value for this setting.	null	No

disableDraw()

If a member with this name exists, and it is a function that returns a boolean variable set to true, we disable the drawing of the visual.

beforeDraw()

If a member with this name exists, it must be a function that executes after receiving data but before the draw event on the visual.

afterDraw()

If a member with this name exists, it must be a function that executes after the draw event on the underlying visual.

afterDataRequest()

If a member with this name exists, it must be a function that executes after creating the request object, but before the data request call. This function accepts the current request object in the arguments, adds your modifications, and returns an updated valid request object. If the function does not return a valid request object, the request is not sent, and the system returns an error.

Extension JS API arcapi functions

Note:

- There are additional functions currently available on the arcapi object, but we do not list them here. These functions were developed for internal use only, and we do not warrant them for use by our clients. Arcadia Data cautions against using them in your customizations because of expected forward compatibility issues.

The extension JS module has access to an arcapi object. Arcadia Data supports the following functions fully:

Table 3: arcapi Functions

Function	Description
arcapi.addScripts()	Adds script elements.
arcapi.addStyles()	Adds style elements.
arcapi.chartId()	Returns the id DOM element.
arcapi.dataResult()	Returns the data result object and gives access to the raw data for plotting the visual.
arcapi.getParameters()	Returns information on all current parameters.
arcapi.getSetting()	Returns the value of the specified setting.
arcapi.sendParameters()	Forwards parameter definitions to other visuals in the app.
arcapi.settings()	Returns information on all current settings.

arcapi.addScripts(filepaths, callback)

Adds script elements for the additional styles specified in the filepaths array. The function callback is invoked after load completes.

Syntax

```
arcapi.addScripts(
  filepaths,
```

```
  callback);
```

Parameters

- *filepaths* An array that specifies the additional JS files to load.
- *callback* Call-back function that is invoked after the load is completed.

Examples

In program code, you can use this API in the following manner:

```
arcapi.addScripts(
  ['https://cdnjs.cloudflare.com/ajax/libs/highcharts/7.0.1/highc
harts.js'],
  function() {
    console.log('highcharts.js has been loaded');
    }
  );
```

This code embeds the following <script> as the last element of the <head> tag in the HTML file:

```
<script
  type="text/javascript"
  src="https://cdnjs.cloudflare.com/ajax/libs/highcharts/7.0.1/hi
ghcharts.js"
  data-loaded="true">
</script>
```

arcapi.addStyles(filepaths, callback)

Adds style elements for the additional styles specified in the filepaths array. The function callback is invoked after load completes.

Syntax

```
arcapi.addStyles(
  filepaths,
  callback);
```

Parameters

- *filepaths* An array that specifies the additional CSS files to load.
- *callback* Callback function that is invoked after the load is completed.

Examples

In program code, you can use this API in the following manner:

```
arcapi.addStyles(
  ['https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4
.2.1/css/bootstrap.css'],
  function() {
    console.log('Twitter Bootstrap has been loaded');
    }
  );
```

This code embeds the following <link> as the last element of the <head> tag in the HTML file:

```
<link
  type="text/css"
  rel="stylesheet"
```

```
  href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstr
ap/4.2.1/css/bootstrap.css"
  data-loaded="true"
>
```

arcapi.chartId()

Returns the id attribute of the DOM element used to draw the visual.

Syntax

```
arcapi.chartId();
```

Usage

```
$("#" + arcapi.chartId())
```

Returns

A jquery selector to the DOM element.

arcapi.dataResult()

Returns the dataResult object that gives the extension access to the raw data used to plot the visual. This object supports the arc.data.result interface.

Syntax

```
arcapi.dataResult();
```

This example shows the result of the arcapi.dataResult() call:

```
f (){return"arc.data.result"}
```

arcapi.getParameters()

Returns a hash object that contains key-value mappings for all current parameters. The key is the parameter name.

Syntax

```
arcapi.getParameters();
```

Example

The following is a typical result of the arcapi.getParameters() command:

```
{
  "productCategory": "Analytics",
  "year": "2019",
  "companies.data": "'Apple', 'Google', 'Arcadia'",
  "companies.exclude": "in"
}
```

See Also

arcapi.sendParameters(params_hash) on page 9

arcapi.getSetting(settingName)

Returns the value of the specified setting of the current JS extension.

Syntax

```
arcapi.getSetting(settingName);
```

Parameters

- *settingName* Name of setting.

Example

The following command retrieves the "Color" setting:

```
arcapi.getSetting("Color");
```

Our example's result follows:

```
"red"
```

See Also

arcapi.settings() on page 9

arcapi.sendParameters(params_hash)

Accepts a hash definition, and sends out the keys and values of the hash as parameters to other visuals in the app.

Syntax

```
arcapi.sendParameters(params_hash);
```

Parameters

- *params_hash* Hash definition.

Example

To pass a parameter with name 'company' and value 'Arcadia Data' to all visuals and filters in the dashboard, issue the following command:

```
arcapi.sendParameters({"company": "'Arcadia Data'"});
```

See Also

arcapi.getParameters() on page 8

arcapi.settings()

Returns information on all current settings for the custom style: an array of objects, with id of a setting, and value of that setting.

The JS extension's settings() function specifies the order of the array.

Syntax

```
arcapi.settings();
```

Example

A typical example output after running the arcapi.settings() function looks similar to this:

```
 [
    {
      id: "Column Name",
      value: "",
```

```
    },
    {
      id: "Max bar width",
      defaultValue: "50",
      value: "50",
    },
    {
      id: "Color",
      defaultValue: "steelblue",
      value: "red",
    }
  ];
```

Extension CSS APIs

Best practice guidelines for CSS customizations:

- Always use a custom selector at the beginning of every rule.
- Can use any class name.
- Can use native DOM element selections.
- Avoid using !important to override existing rules.

Table 4: CSS Classes

Class	Description
arc-dashboard-title-container	Container for the title of the dashboard.
arc-dashboard-title	The title of the dashboard.
arc-dashboard-subtitle-container	Container for the subtitle of the dashboard.
arc-dashboard-subtitle	The subtitle of the dashboard.
arc-app-container	Container class for dashboards. One exists on the page when rendering the dashboard.
arc-viz-container	Container class for visuals. One exists on the page for each visual.
arc-viz-type-visualtype	Container class for a specific visual type. For example, tables have an arc-viz-type -table class.
arc-viz-title-container	Container class for a div that surrounds the title of the visual. Commonly used to center the title within this container.
arc-viz-subtitle-container	Container class for a div that surrounds the subtitle of the visual. Commonly used to center the subtitle within this container.
arc-viz-title	Class applied to the title of the visual.
arc-viz-subtitle	Class applied to the subtitle of the visual.
arc-viz-axis	Class applied to the axis of the visual.
arc-viz-legend	Class applied to the color or mark legend of the visual.
arc-viz-axis-label	Class applied to the axis label of the visual.