Advanced API

Overview

This API will only be available on pages that have the included the “JsAPI/v3” JS file into their page. See example:

```
<script src="https://pathToYourYellowfinServer/JsAPI/v3"></script>
```


This API will not be available in Dashboard Code Mode.

The Advanced API contains a number of functions to assist with loading the Yellowfin APIs in your own web page. There are functions to load the report and dashboard loader APIs which will allow you to load Dashboard and Reports.

There is also a function to assist with session management.

init()

Returns

Promise

Description

Initialises the Base API. This is called when the API is included into a page. Returns a promise to indicate when this has finished loading, any usage of the API should wait until this promise has resolved.

Example

Include the Yellowfin Report API on the page and then load a report after the **init()** promise has resolved:

```
<script src="https://pathToYourYellowfinServer/JsAPI/v3"></script>
<div id="reportDiv"></div>
<script>
yellowfin.init().then(() => {
        //The Yellowfin base API has now loaded. Now load a report
        yellowfin.loadReport({
                reportId: 'c83357db-8aef-4ec7-ab72-fce34de9ee77',
                element: document.querySelector('div#reportDiv'),
         });

});
</script>
```

logoff()

Returns

Nothing

Description

Ends the current user session.

yellowfin.newSession(token, org);

Returns

Promise

Description

This function logs out the current user and sends a request to the Yellowfin JS API to login a new user using the passed webservice session token. Along with the session token, this function accepts an optional client org ID so that the user can be logged into a specific client org.

When the login action has successfully completed, the function will return a promise, and any reports or dashboards on the page will be re-run.

loadReport(reportOptions)

Returns

Promise

Description

Waits for the *init()* promise to load and then loads a report with the passed options. Once the report has been loaded it will be appended to the passed element and the report will begin to run.

When the promise is resolved a Report object will be passed to the ***.then()*** functions. The promise will be resolved just before the report starts running.

There are a number of options that will be accepted as part of the options object:

reportId (required)

Type: String

ReportUUID of the report we wish to load. This is the PublishUUID on the ReportHeader table.

element *(*required)

Type: DOM Element, Selector String

Element that you wish to render the report into. Can either be a DOM element or a selector that will be used to select the element from the page.

If no width or height is passed the report will expand to the dimensions of the element.

width

Type: Number

Width to render the report in pixels.

height

Type: Number

Height to render the report in pixels.

filterValues

Type: Array[Object]

Array of Objects that will be used to apply the initial filter values.

See the FiltersAPI to understand the filter values.

The objects should contain the following information:

- **filterId** - UUID or Name of the filter you wish to set.
- **valueOne** - The first value of the filter.
- **valueTwo** - The second value of the filter, only has an affect on Between and Not Between filter types.
- **valueList** - Array of values to apply to a list filter.

```
[{ //Set a between filter
        filterId: '1c1e0878-3bd6-402b-bad8-98202e6b8fb1',
        valueOne: 15,
        valueTwo: 35
}]
```

instanceName

Type: String

Identifier for this instance of the report. If this property is not defined, a random id will be generated. This means that if you include the same report multiple times without defining an instance name, it will always be a separate instance of that report.

If you do set this, it allows you to load multiple views of the same instance of the report.

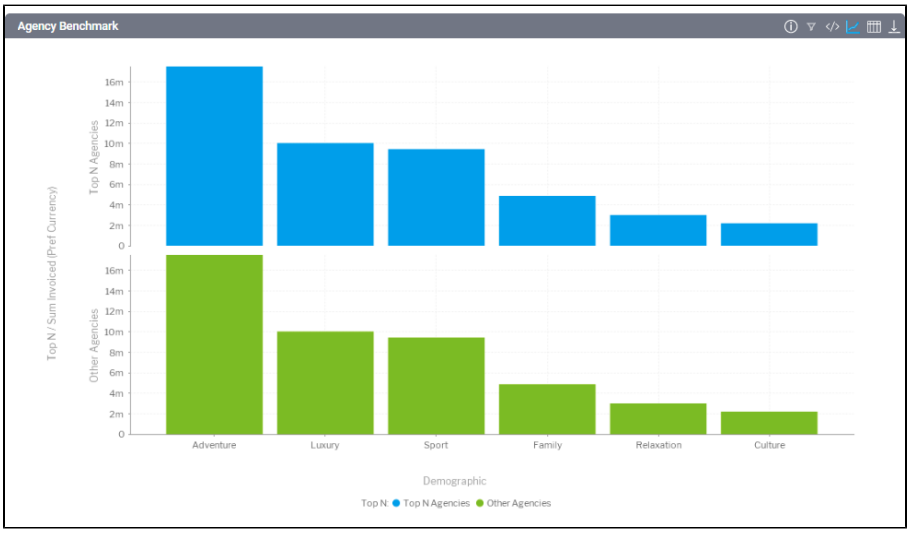
```
//Load the report into two different elements
yellowfin.loadReport({
        reportId:'9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#firstReport'),
        instanceName: 'firstReport'
});

yellowfin.loadReport({
        reportId:'9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#secondReport'),
        instanceName: 'firstReport'
});
```

showToolbar

Type: Boolean

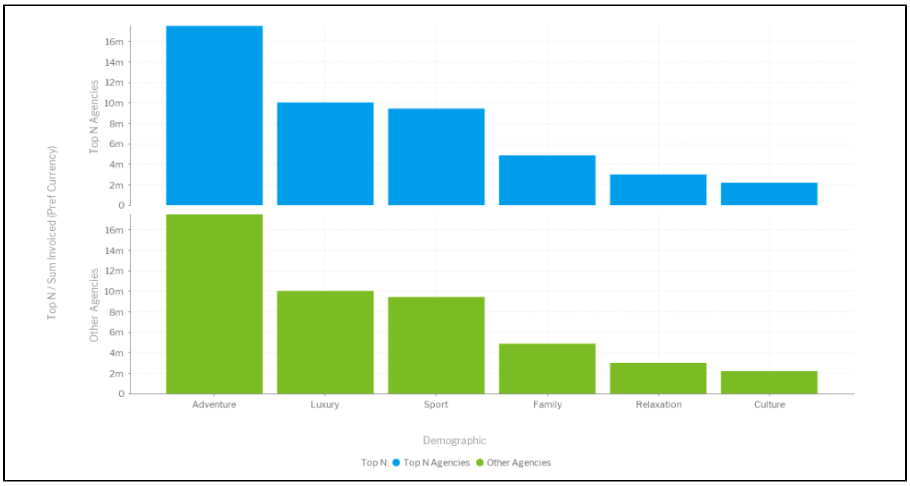
Determines whether or not to show the reports toolbar.

```
//Load the report with showToolbar not defined
yellowfin.loadReport({
        reportId:'9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv');
});
//Load the report with showToolbar set to true
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showToolbar: true
});
```



Or set the showToolbar property to false:

```
//Load the report with showToolbar set to false
yellowfin.loadReport({
    reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
    element: document.querySelector('div#reportDiv'),
    showToolbar: false
});
```



showTitle

Type: Boolean

Determines whether or not to show the report title in the report's toolbar. Default: true.

If the **showTitle** option is set to false:

```
//Load the report with showTitle option set to false
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showTitle: false
});
```

showInfo

Type: Boolean

Determines whether or not to show the info option in the report's toolbar. Default: true

If this option is set to false:


```
//Load the report with showInfo option set to false
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showInfo: false
});
```

showFilter

Type: Boolean

Determines whether or not to show the filters option in the reports toolbar. Default is true.

If this option is set to false:


```
//Load the report with showFilter option set to false
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showFilter: false
});
```

showExport

Type: Boolean

Determine whether or not the export option will be displayed in the report toolbar. Export options allow a report to be exported to an external file format such as csv, xls, pdf, txt and doc.

With the **showExport** parameter set to false, the export icon in the toolbar will be removed:

```
//Load the report with showExport option set to false
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showExport: false
});
```

showShare

Type: Boolean

Determines whether or not to show the share option in the reports toolbar. Default is true.

```
//Load the report with showShare option set to false
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showShare: false
});
```

showDisplayToggle

Type: Boolean

Determines whether or not to show the reports chart/table options in the reports toolbar. Default is true.

If this option is set to false:


```
//Load the report with this option set to false
yellowfin.loadReport({
        reportId: '9617ada1-28bc-42ef-9c3f-8b40d3d1ae61',
        element: document.querySelector('div#reportDiv'),
        showReportDisplayToggle: false
});
```

Example

Load the report and then print its name:

```
yellowfin.loadReport({
         reportId: aReportId,
        element: elementToRenderTo
}).then(report => {

         //Some code to execute when the report has loaded

         console.log(report.name + ' has finished loading');
});
```

Interactions

The interactions object enables you to define which interactions should be available on a particular visualization of a report. This lets you tell the Yellowfin renderer which functionality you want the user to be able to use within that visualization.

If there are multiple elements representing the same report and you wish to disable an interaction, you must disable it on all the elements you are displaying. For example, if you disabled drill down on the first element and then did not on the second, the user would still be able to drill down on the second element.

By default, any interaction that is not explicitly set to false is treated as enabled and will display if the report allows it.

drillDown

Allows drill down functionality to be disabled.

drillAnywhere

Allows drill anywhere functionality to be disabled.

drillThrough

Allows drill through functionality to be disabled.

unitSelection

Allows unit selection functionality to be disabled.

brushing

Allows brushing functionality to be disabled.

timeSlider

Allows the timeSlider functionality to be disabled.

drillBreadcrumbs

Allows drill breadcrumb functionality to be disabled

seriesSelection

Allows seriesSelection functionality to be disabled

annotations

Allows annotations to be disabled.

Example

To disable all user interactivity on the report when using loadReport:

```
yellowfin.loadReport({
    reportId: 'a report id',
    interactions: {
         drillDown: false,
        drillAnywhere: false,
        drillThrough: false,
        unitSelection: false,
        brushing: false,
        timeSlider: false,
        drillBreadcrumbs: false,
        seriesSelection: false,
        annotations: false
    },
    element: document.querySelector('#reportElement')
}).
```

top

---

loadDashboard(dashboardOptions)

Returns

Promise

Description

Waits for the *init()* promise to load and then loads a dashboard with the passed options. Once the dashboard has been loaded it will be appended to the passed element and the content on the dashboard will begin to run.

There are a number of parameters that can be included in the **dashboardOptions** object:

dashboardUUID (required)

Type: String

The PublishUUID of the dashboard that you wish to load.

element (required)

Type: DOM Element, Selector String

Element that you wish to render the dashboard into. Can either be a DOM element or a selector that will be used to select the first instance of that selector from the page.

If no width or height is passed the dashboard will fill the dimensions of the element.

filterValues

Type: Array[Object]

Array of Objects that will be used to apply the dashboards initial filter values.

See the FiltersAPI to understand the filter values

The objects should contain the following information:

- **filterId** - UUID or Name of the filter you wish to set.
- **valueOne** - The first value of the filter.
- **valueTwo** - The second value of the filter, only has an affect on *Between* and *Not Between* filter types.
- **valueList** - Array of values to apply to a list filter.

```
[{ //Set a between filter
        filterId: '1c1e0878-3bd6-402b-bad8-98202e6b8fb1',
        valueOne: 15,
        valueTwo: 35
}]
```

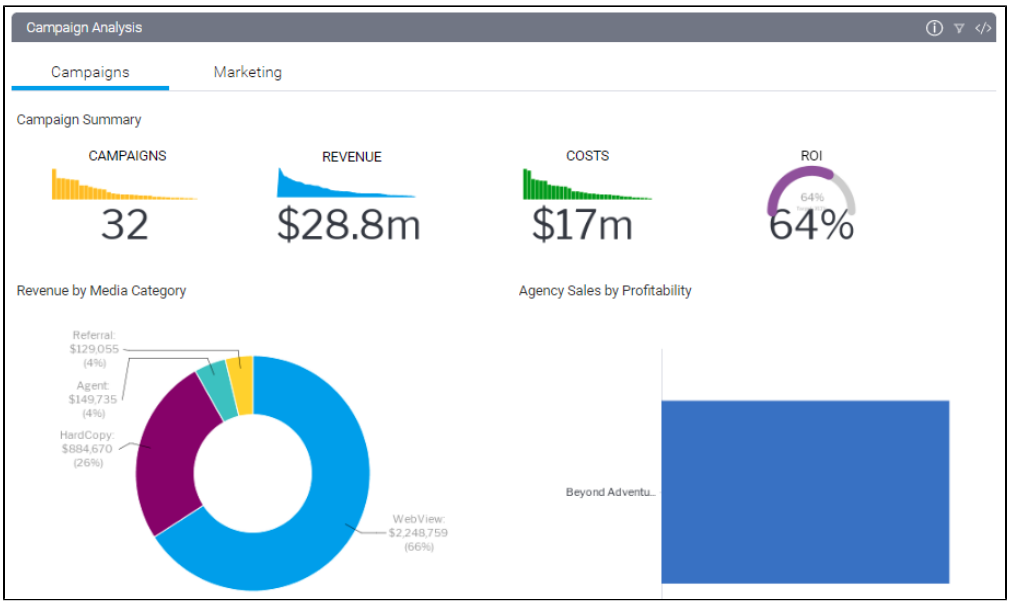
showToolbar

Type: Boolean

Determines whether or not to show the dashboard's toolbar. Default is true.

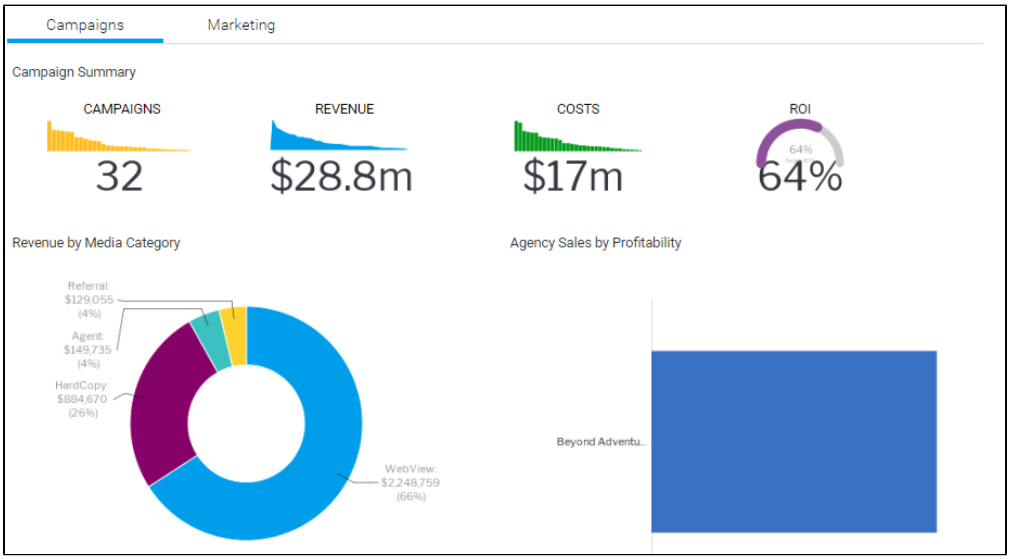
```
//Load the dashboard with showToolbar not defined
yellowfin.loadDashboard({
        dashboardUUID:'1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv');
});
//Load the dashboard with showToolbar set to true
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        showToolbar: true
});
```

The following is a dashboard with the toolbar showing:



But if the **showToolbar** property is set to false:

```
//Load the dashboard with showToolbar set to false
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        showToolbar: false
});
```



showInfo

Type: Boolean

Determines whether or not to show the info option in the dashboard's toolbar. Default: true.

If set to false:

```
//Load the dashboard with showInfo option set to false
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        showInfo: false
});
```

showShare

Type: Boolean

Determines whether or not to display the share option in the dashboard's toolbar. Default is true.

If set to false:


```
//Load the dashboard with showShare option set to false
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        showShare: false
});
```

showFilters

Type: Boolean

Determines whether or not to show the filters option in the dashboard's toolbar. Default is true.

If set to false:


```
//Load the dashboard with showFilter option set to false
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        showFilter: false
});
```

scaleCanvasTabs

Type: Boolean

Determines whether or not a canvas tab should be scaled. Default is true.

If this is set to false, then the canvas tab will appear at the exact size it was created. This setting has no effect on sub-tabs which don't contain canvas tabs.

```
//Load the dashboard with scaleCanvasTabs option set to false
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        scaleCanvasTabs: false
});
```

showGlobalContentContainer

Determines whether or not to show the global content containers when rendering the dashboard.

```
//Load the dashboard with showGlobalContentContainer option set to false
yellowfin.loadDashboard({
        dashboardUUID: '1e68d9cc-fa5a-44e2-816d-782aa40ceeae',
        element: document.querySelector('div#dashboardDiv'),
        showGlobalContentContainer: false
});
```

top

Embedding stories with the Advanced API

From Yellowfin 9.4, you can embed a story with the publish UUID of the story (see the JS API to embed Yellowfin content wiki page for more info).

Using the Advanced API toolset expands the options available to you when embedding a story, so that you can adjust the look and feel of the story to suit your needs. You may wish to hide contributors or limit the width of the embedded story, and the Advanced API lets you do this.

Advanced API

A new object will be added to the yellowfin object — *stories*. The code below is an example of how a story would typically be loaded using the Advanced API.

```
<script src="http://yellowfinServer/JsAPI/v3"></script>
<div id="myStoryDiv"></div>
<script>
    yellowfin.stories.loadStory({
         storyUUID: 'a-story-uuid', //This should be the story's publish uuid
         element: document.querySelector('#myStoryDiv') //The div to render the story into
    });
</script>
```

yellowfin.stories

Currently, yellowfin.stories can only contain one function as shown above — `loadStory` — which will load the story. However, the `loadStory` function can take a number of options. The function and its options are outlined below.

loadStory(options);

Details

Returns: Promise

Promise parameters: storyAPI

Description

This function loads the story associated with the passed `options.storyUUID`, and returns a promise that is resolved once complete.

The resolved promise will pass a `storyAPI` object.

The possible values that can be added to the options object are below.

storyUUID

Details

Embed link option: ?StoryUUID=uuid (This is the PublishUUID for the story to be embedded: if this appears later in your query string, don't forget to change the ? to a &.)

Description

This option is required: if it is not passed, the loadStory call will fail.

element

There is no embed link option for this.

This is the element that we wish to render the story to.

showToolbar

Details

Embed link parameter: &showToolbar=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the JS API toolbar is shown for the embedded story.

showInfo

Details

Embed link parameter: &showInfo=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's info dropdown is visible.

showExport

Details

Embed link parameter: &showExport=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's PDF export options are visible to the story viewer.

showBannerImage

Details

Embed link parameter: &showBannerImage=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's banner image is shown.

showHeader

Details

Embed link parameter: &showHeader=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines if the embedded story's header is shown.

A story's header includes the fields for author and date, and the counters for likes and reads. To display any of these, first include this `showHeader` option, and set it to true. If this option is set to true and the four elements it can contain are switched off, a blank banner will be displayed.

showAuthor

Details

Embed link parameter: &showAuthor=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's author is shown. Even if it's set to true, it will only be displayed if the `showHeader` option is also present and set to true.

showDateContainer

Details

Embed link parameter: &showDateContainer=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's publication date is shown. Even if it's set to true, it will only be displayed if the `showHeader` option is also present and set to true.

showLikeButton

Details

Embed link parameter: &showLikeButton=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the story's 'Like' button is shown. Even if it's set to true, it will only be displayed if the `showHeader` option is also present and set to true. Guests who are not authenticated will not be able to use this button.

showReadBy

Details

Embed link parameter: &showReadBy=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the list of people who have read the embedded story is shown. Even if it's set to true, it will only be displayed if the `showHeader` option is also present and set to true.

showStoryTitle

Details

Embed link parameter: &showStoryTitle=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's title is displayed.

showFooter

Details

Embed link parameter: &showFooter=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the embedded story's footer is displayed.

The story footer includes contributors. To display the list of contributors, first include this `showFooter` option, and set it to true.

showContributors

Details

Embed link parameter: &showContributors=true (Or if it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: true

Description

This option determines whether the list of story contributors is shown when embedded. Even if it's set to true, it will only be displayed if the `showFooter` option is also present and set to true.

width

Details

Embed link parameter: &width=xxx (where xxx is a number. If it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: element width

Description

This option sets the total width of the embedded story.

storyWidth

Details

Embed link parameter: &storyWidth=xxx (Where xxx is a number. If it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: 3/4 of *width*

Description

This option sets the full width of the embedded story's body (which doesn't include the banner). The width of most of the story's contents will be 200px less, unless it is set to display at full width (eg, an image in wide mode).

bannerImageHeight

Details

Embed link parameter: &bannerImageHeight=xxx (Where xxx is a number. If it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: 300px

Description

This option sets the custom height that should be applied to the banner image container. For best results, we recommend that you also set the banner width if you use this setting.

bannerImageWidth

Details

Embed link parameter: &bannerImageWidth=xxx (Where xxx is a number. If it's the first element in the query string, change the prefix from *&* to *?* like any other query string.)

Default value: Value of *width*

Description

This option sets the custom width that should be applied to the banner image. For best results, we recommend that you also set the banner height if you use this setting.

loadStoryAPI()

Loads the StoryAPI so that we can load stories, returns a promise that is resolved when the API loads.

loadStory(options)

**This is just a pass through option for yellowfin.stories.loadReport the options that are passed to it should be the same, it just ensures that the stories object is loaded before attempting to load a story.

Functions available in the StoryAPI

- likeStory()
- unlikeStory()
- toggleLike(shouldLike:boolean => Default inverse of current value)
- getCurrentLikeStatus(successFunction: function to call one the current status is received) => Get status from the server
- storyContributors()
- storyReaders()

Example:

```
src="<%=YF URL%>/JsAPI/v3?StoryUUID=<%=Story Publish UUID%>&showHeader=false&bannerImageWidth=500"></script>
```

Example Advanced API :

HTML:

```
<script src="<%=YF URL%>/JsAPI/v3"></script>

<div id="storyContainer"></div>
```

JavaScript:

```
window.yellowfin.loadStoryAPI().then(() => {
    window.yellowfin.stories.loadStory(
{         storyUUID: '<%=Story Publish UUID%>',          element: 'div#storyContainer'     }
).then((storyAPI) =>
{         console.log(storyAPI);         window.storyAPI = storyAPI;     }
);
});
```

top

Set up Guided NLQ via the Advanced API

The JS API provides two means for setting up Guided NLQ:

- via the Advanced API (discussed below); and
- via the embed link.

Advanced API

The Advanced API introduces many complex objects and allows different methods of setting up the Guided NLQ view.

The simple approach is:

```
yellowfin.loadNLQ();
```

This will load the Guided NLQ lightbox into the embedded page and allow it to function exactly as it does natively within the application. This approach takes the following settings as defaults:

```
contentIntegrationOptions: {
    controls: ['SAVE']
}
 When you called yellowfin.loadNLQ(); you are essentially calling:
yellowfin.loadNLQ({
   contentIntegrationOptions: {
      controls: ['SAVE']
   }
});
```

You can define the following options when loading Guided NLQ using the Advanced API.

element

Details

Type: DOM Element Or String

Description

Use this option to define the element where Guided NLQ UI will be rendered. If you pass a string, the JS API will use document.querySelector(stringValue) to attempt to fetch the element from the page.

```
yellowfin.loadNLQ({
    element: document.querySelector('#myElement')
});
```

The Guided NLQ item will then load into that element.

If no string or DOM element is passed, the Guided NLQ UI will be loaded in a lightbox .

Note: When defining an element in this manner, the Close button will be hidden in the NLQ UI.

contentIntegration

Details

Type: Object

Description

Define which content integration options to make available to the user.

This object has a number of sub-objects to be defined (see below for details and defaults). This lets you to define an array of integration controls. In the example below, the controls SAVE and ADD_TO have been included in the array.

controls

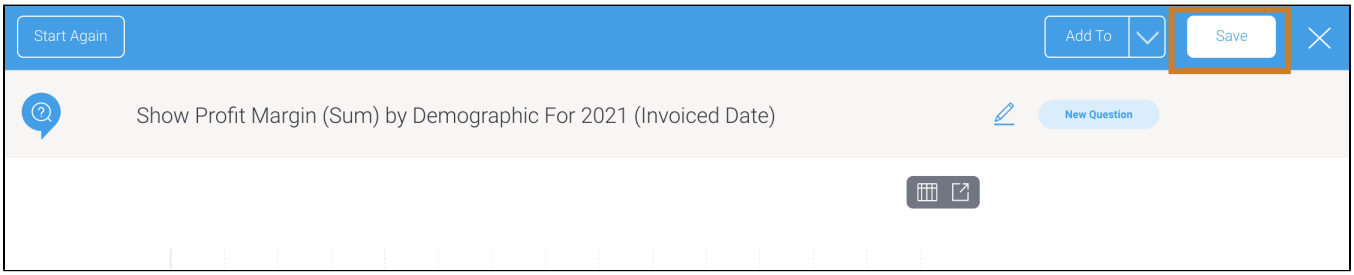
Type: String[]

```
yellowfin.loadNLQ({
    contentIntegrationOptions: {
        controls: ['SAVE', 'ADD_TO']
    }
});
```

The values that can be passed in this array are:

SAVE control

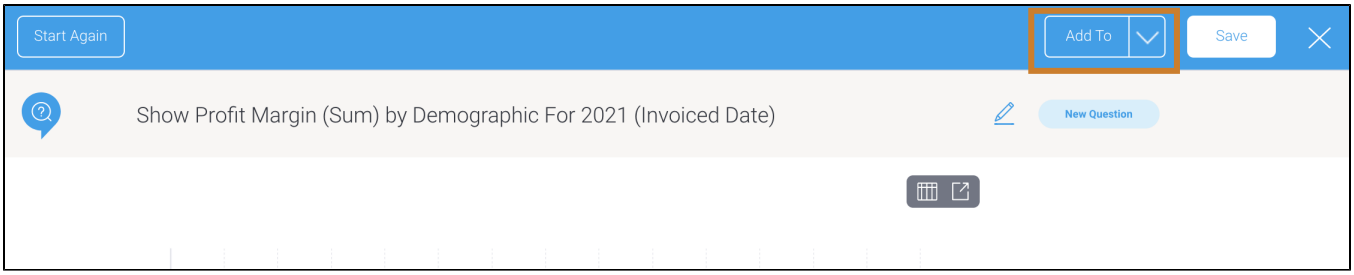
This SAVE control shows the Save button after the query has completed. If the user does not have access to create a report, this option will not be shown, even if SAVE is defined.




```
yellowfin.loadNLQ({
    contentIntegrationOptions: {
        controls: ['SAVE']
    }
})
```

ADD_TO control

The ADD_TO control shows the Add To dropdown, where users can add their Guided NLQ answer to stories, dashboards and presentations, if their access allows.




```
yellowfin.loadNLQ({
    contentIntegrationOptions: {
        controls: ['SAVE', 'ADD_TO']
    }
});
```

A Guided NLQ answer is actually a report. Therefore, the ADD_TO control requires the SAVE control to be defined along with it. If you pass ADD_TO without SAVE, neither option will be shown.

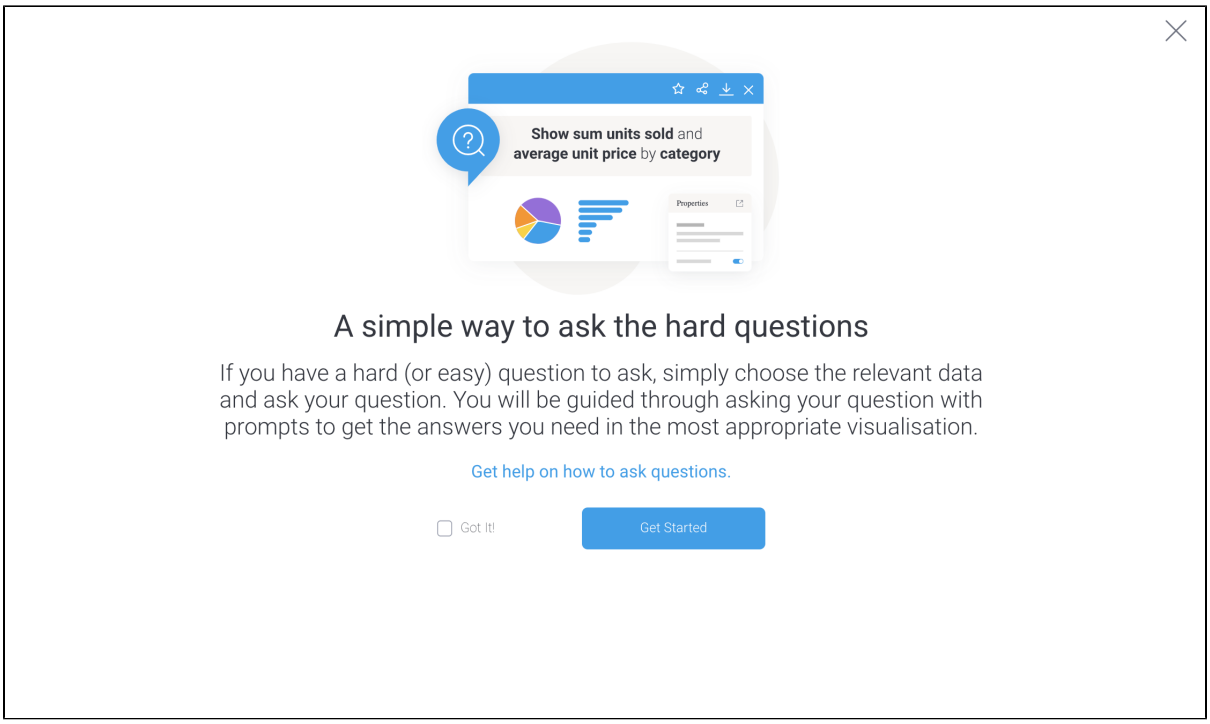
showWelcome

Details

Type: String

Description

This string determines whether or not to show the Welcome splash screen.



There are three possible options for this:

ALWAYS

This will show the Welcome splash screen every time the NLQ view is initialised, regardless of the user checking the 'Got It' checkbox..

NEVER

This will never show the Welcome splash screen.

NORMAL

This will show the Welcome splash screen every time a user loads Guided NLQ until they check the 'Got It' checkbox. This is the default value. If no value is passed, the default value is NORMAL.

popup

Details

Type: Boolean

Description

If `popup` is set to true, the NLQ UI will be shown in a lightbox, regardless of the value of the `element` option. If the `element` option is set, it will be ignored.

If `popup` is set to false and the `element` option is not null, the NLQ UI will not be shown in a lightbox.

If `popup` is set to false and the `element` option is null, the NLQ UI will still be shown in a lightbox.

viewUUID

Details

Type: String

Description

This string is the UUID of the Guided NLQ-enabled view to be made available to the user. If this is passed and the view is valid for that user, the screen where the user would select a view will be skipped. If the passed UUID doesn't match a view that the user has access to, the screen for view selection will be displayed.

Events

The Guided NLQ object will trigger a number of events based on user actions and processes completing on the Yellowfin server. You can listen for the events using the syntax:

```
yellowfin.loadNLQ().then(nlq => {
nlq.addEventListener('eventName', eventCallbackFunction);
});
```

The following events may be triggered:

reportCreated

This is triggered when a valid visualization is created. This will trigger an event containing the Guided NLQ object's report object as well as any previous value of this, if it exists. In most cases, the previousValue object will be null. However, if a user were to click Start Again, it is possible the previous version would be overwritten when *reportCreated* is triggered.

The following data is provided as part of the event

- **report**: The Report API that has been generated by Guided NLQ

runQueryStart

This is triggered when the user clicks the Ask a Question button.

The object doesn't provide any information about the question — just that a question has been asked.

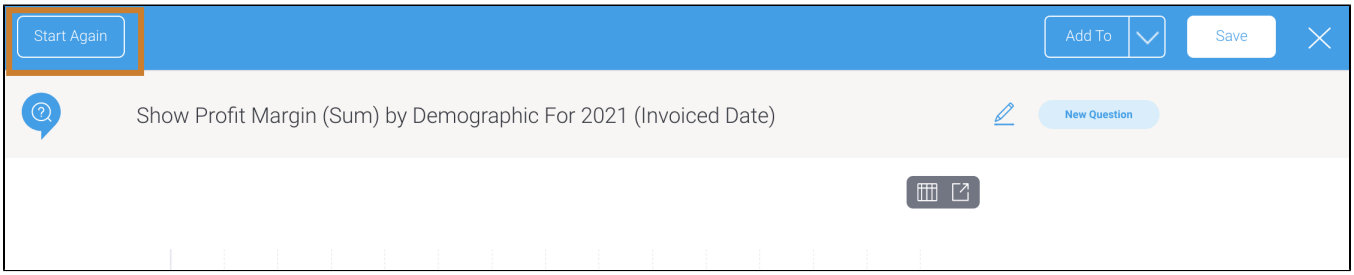
runStatusUpdated

This is triggered when the user adds or removes a part to the question. It will provide information about whether or not the question is in a runnable state:

- **isRunnable** (boolean): True if the question is in a runnable state; false if not.
- **detailedMessage** (String): Message detailing why the question is not runnable. This string will be empty if isRunnable is true.
- **Error** (Object): Error object containing the error_id and the message about why the report is not runnable. If the question is runnable, this will be false.

reset

This is triggered when the user clicks the Start Again button.



viewChanged

This is triggered when the user selects a view for asking a question. It will provide the following information:

- **uuid** (String): UUID of the view that has been selected.
- **previous** (String): UUID of the previous view that was selected. If there was no previous value, this will be null.

reportSaved

This is triggered when the user has accessed the Save as Report dialog box (from the Save button) popup and successfully saves the report. It will provide the following information:

- **chartId** (Integer): The ID of the chart that was generated.
- **description** (String): The description of the draft report that has just been saved.
- **isTable** (Boolean): Whether or not the generated report has a table.
- **name** (String): Name of the report.
- **reportId** (Integer): Integer ID of the report.

- **reportUUID** (String): UUID of the report.

The reportUUID should be used if you wish to load the report and display it elsewhere on the page.

This event is preventable. If it is prevented, the “Do you want to go to report” popup will not be displayed, but the report will still be saved in Yellowfin.

top
