JS API in Code Mode

- Getting started in Code Mode
  - How to Turn Code Mode On
    - Role Function
    - Configuration
- Using Code Mode
  - HTML
    - Common Attributes
      - width
      - height
      - left
      - Top
      - name
      - publishUUID (new since Yellowfin 9.5)
      - widgetUUID (deprecated from Yellowfin 9.5)
  - JS
    - Functions
      - onRender
      - onRemove
    - How to get APIs
    - How to get Reports
    - Updating Element Attributes from JS
  - CSS
  - Examples
    - Update reports visualisation
    - Using the report API Dataset to build a Custom Table

Getting started in Code Mode

Code Mode can be used on Canvas dashboards to enhance the capabilities of that dashboard.

How to Turn Code Mode On

Role Function

The **"Code Editor"** role function needs to be enabled, this can be found under **"Dashboard"** in the role settings page.

Configuration

As well as turning on the role function, the **"Dashboard Code Mode"** option needs to be turned on in the Admin Console. This can be found at Administrati on -> Configuration -> System -> Security.

If this setting is turned *off* while you have a Code Mode dashboard, the Code Mode dashboard will no longer run.

Using Code Mode

Once Code Mode has been enabled and you edit a dashboard, you will see this toggle in the dashboard toolbar:

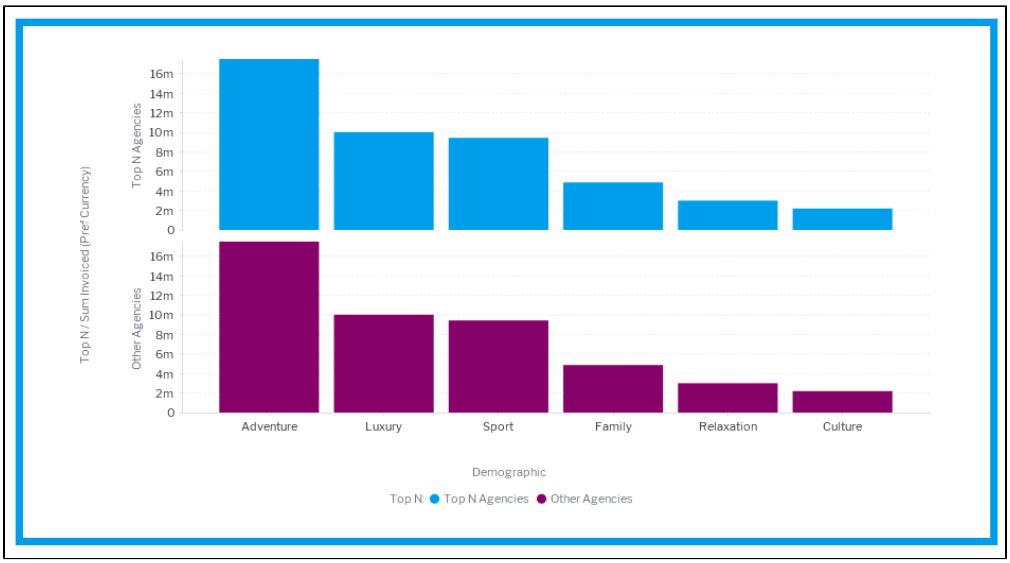
Clicking on **"Code"** will take you to Code Mode.

When switching into Code Mode, you will see 3 tabs: the HTML, JS and CSS tabs.

HTML

The HTML tab shows the current canvas tab represented in HTML. Each item on a canvas has a corresponding HTML tag.

The following canvas has two elements on it — a report and a rectangle used as a border:



The HTML for this canvas is as follows:

```
<canvas-area xmlns="http://www.w3.org/1999/xhtml" canvas-uuid="aee7c574-3ad3-4ac0-a366-7810861d91eb">
<!-- This the element tag for the Report-->
        <report-output publish-uuid="ffe09245-8739-416c-b039-05fff6427242" report-uuid="c83357db-8aef-4ec7-ab72-fce34de9ee77" height="475" top="
46" left="54" name="Agency Benchmark" width="897" display-type="CHART" chart-uuid="a9aba052-51b5-47b1-b311-2c0cc3abd932" on-click="none"
style="z-index: 3;"></report-output>
<!-- Element tag for the rectangle -->
                <canvas-rectangle publish-uuid="d0fafeb9-57a8-4a21-ab6c-844501ebf50b" width="927" height="496" top="35" left="36" rotation="0" name=
"Rectangle" stroke-color="#009eec" fill-color="#ffffff" on-click="none" style="z-index: 2" stroke-width="14" fill-opacity="100"></canvas-rectangle>
</canvas-area>
```

Any changes that are made to the attributes of these tags in the HTML tab will be synchronized to the Canvas visual editor. If you remove the *chart-uuid* att ribute and change *display-type* to “REPORT”, this will be updated in the visual editor.

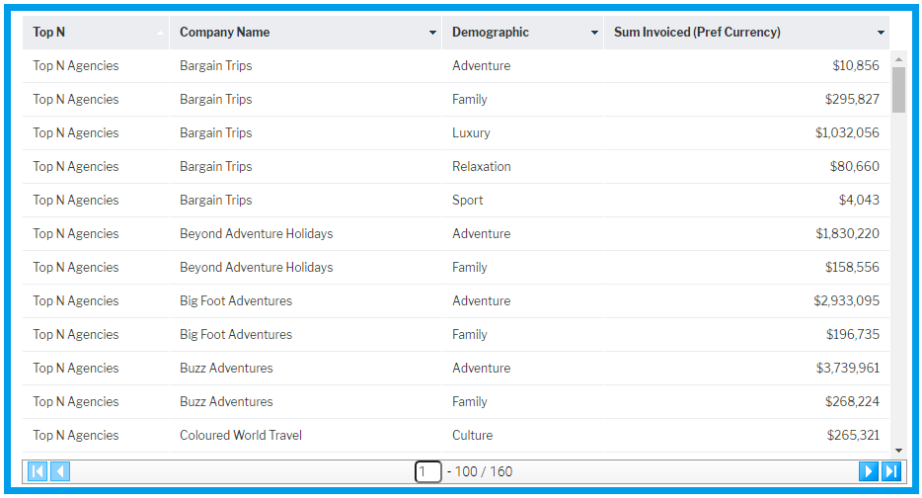
> From Yellowfin 9.5, publish-uuid replaces the older widget-uuid. This page has been updated to reflect the updated UUID. For Yellowfin users on older versions, we recommend you upgrade, or use the name attribute in place of publish-uuid in the interim.

Code

```
        <report-output publish-uuid="ffe09245-8739-416c-b039-05fff6427242" report-uuid="c83357db-8aef-4ec7-ab72-fce34de9ee77" height="475" top="
46" left="54" name="Agency Benchmark" width="897" display-type="REPORT" on-click="none" style="z-index: 3;"></report-output>
```

> From Yellowfin 9.5, publish-uuid replaces the older widget-uuid. This page has been updated to reflect the updated UUID. For Yellowfin users on older versions, we recommend you upgrade, or use the name attribute in place of publish-uuid in the interim.

Visual



Top N	Company Name	Demographic	Sum Invoiced (Pref Currency)
Top N Agencies	Bargain Trips	Adventure	$10,856
Top N Agencies	Bargain Trips	Family	$295,827
Top N Agencies	Bargain Trips	Luxury	$1,032,056
Top N Agencies	Bargain Trips	Relaxation	$80,660
Top N Agencies	Bargain Trips	Sport	$4,043
Top N Agencies	Beyond Adventure Holidays	Adventure	$1,830,220
Top N Agencies	Beyond Adventure Holidays	Family	$158,556
Top N Agencies	Big Foot Adventures	Adventure	$2,933,095
Top N Agencies	Big Foot Adventures	Family	$196,735
Top N Agencies	Buzz Adventures	Adventure	$3,739,961
Top N Agencies	Buzz Adventures	Family	$268,224
Top N Agencies	Coloured World Travel	Culture	$265,321

1 - 100 / 160

Common Attributes

While many of the elements that can be added to a canvas will have different attributes, they will all share the following attributes.

width

Initial width of the widget in pixels.

height

Initial height of the widget in pixels.

left

Initial left position of the widget in pixels.

Top

Initial top position of the widget in pixels.

name

Name of the widget, this can be used as an identifier to easily select the widget from the DOM.

publishUUID (new since Yellowfin 9.5)

Unique ID for the widget. This can be used for selecting items from the canvas.

widgetUUID (deprecated from Yellowfin 9.5)

Unique ID for the widget. This should not be used for selecting items from the canvas. We recommend using the name attribute in place of widget-uuid if you're using an earlier version of Yellowfin 9.

JS

The JS tab allows you to write JavaScript that will be executed when a user navigates to your tab. When you first navigate to this tab, it will contain the following code:

```
// Declare variables or insert code here

/**
* Called when the canvas and all its child elements are rendered and attached into the page.
*/
this.onRender = function () {
        // Insert your code here. This is an ideal place for setting up event listeners
};

/**
* Called when the canvas and all its child elements are removed from the page.
*/
this.onRemove = function () {
        // Insert your code here. This is an ideal place for removing event listeners
};
```

Functions

onRender

Description

This function is called when a SubTab is rendered on the page and will be called every time the SubTab becomes visible again. This is a good spot to add any event listeners to the page that you might require.

This can be added to the Code Mode object by using:

```
this.onRender = function() { //Your logic here }
```

onRemove

Description

This is called when the SubTab is removed from the page, this will happen when a user navigates to a different tab.

This method should be used for removing any event listeners or objects that you don't wish to persist when moving to a new tab.

Example

```
let subTabAPI = this.apis.subtab;

this.bodyClick = function() {
        console.log('A click was recorded on ' + subTabAPI.name);
});

this.onRender = function() {
       document.body.addEventListener('click', this.bodyClick);
});

this.onRemove = function() {
       document.body.removeEventListener('click', this.bodyClick);
};
```

How to get APIs

The "this" object will contain an "apis" parameter. This object will contain the following objects:

- canvas
- dashboard
- subtab
- filters

These can then be retrieved like so:

```
this.apis.canvas;
this.apis.subtab;
this.apis.dashboard;
this.apis.filters
```

How to get Reports

One of the main advantages of using Code Mode is you can access the Reports APIs, however the ReportAPIs are not passed as part of the *this.apis* obje ct as they are loaded dynamically as required. Once a report is going to be rendered the Dashboard will create its Report API.

If you are attempting to fetch the ReportAPI for a report that is visible on the current SubTab, you can use the CanvasAPI ***select*** or ***selectAll*** functions to select the report element on the page and retrieve the API from there. The *select* and *selectAll* are functions that will create a selector to be passed into the canvas elements ***querySelector*** and ***querySelectorAll*** functions.

In this example, we've given our report the name "report_1". We can now use the canvasAPI *select* function to select that element, and then get the reportAPI from it.

```
let reportElement = this.apis.canvas.select('report_1');
reportElement.onReportLoad.then(() => { //onReportLoad is a promise that is resolved once the reportAPI for that report element has finished loading.
	let reportAPI = reportElement.reportAPI; //This is now the ReportAPI.
});
```

The second approach is using a ***dashboard getAllReports()*** or ***getReportsForSubTab()*** combined with the ***dashboard.loadReport()*** function. This is outlined in the DashboardAPI.

Updating Element Attributes from JS

You can update an element's HTML attributes from your code in the JS tab, however these changes will not be saved to the database like changing them in the HTML tab would. It will however cause that element to update.

CSS

The CSS tab contains any styling that you wish to apply to your SubTab. Any CSS applied here will have no effect on the rest of the page. When a user navigates away from the SubTab, this CSS will be removed from the page.

Examples

Update reports visualisation

This example takes a report that is on the dashboard and toggles its visualisation every 5 seconds.

HTML

```
<canvas-area xmlns="http://www.w3.org/1999/xhtml" canvas-uuid="aee7c574-3ad3-4ac0-a366-7810861d91eb">
	<report-output publish-uuid="ffe09245-8739-416c-b039-05fff6427242" report-uuid="c83357db-8aef-4ec7-ab72-fce34de9ee77" height="475" top="46" left="54" name="report_1" width="897" display-type="CHART" chart-uuid="a9aba052-51b5-47b1-b311-2c0cc3abd932" on-click="none" style="z-index: 3;"></report-output>
	<canvas-rectangle publish-uuid="d0fafeb9-57a8-4a21-ab6c-844501ebf50b" width="927" height="496" top="35" left="36" rotation="0" name="Rectangle" stroke-width="14" stroke-color="#009eec" fill-opacity="100" fill-color="#ffffff" on-click="none" style="z-index: 2;"></canvas-rectangle>
</canvas-area>
```

From Yellowfin 9.5, publish-uuid replaces the older widget-uuid. This page has been updated to reflect the updated UUID. For Yellowfin users on older versions, we recommend you upgrade, or use the name attribute in place of publish-uuid in the interim.

JS

```
this.onRender = function () {
	//Select the report_1 element from the page canvas through the canvas api
	let report1 = this.apis.canvas.select('report_1');

	//The report will begin loading once it is added to the DOM, so it is possible that at this point it hasn't completed yet. So wait for the onReportLoad promise to resolve.
	report1.onReportLoad.then(() => {
		let changeDisplayType = () => {
			let currentDisplay = report1.getAttribute('display-type');
			if(currentDisplay === "CHART") {
				currentDisplay = "REPORT";
			} else {
				currentDisplay = "CHART";
			}

			report1.setAttribute('display-type', currentDisplay);
		}
		this.reportChangeTimeout = setInterval(changeDisplayType, 5000);
	});
};

this.onRemove = function () {
	//The SubTab is being removed from the page, we want to stop triggering the reportChange
	clearTimeout(this.reportChangeTimeout);
};
```

Using the report API Dataset to build a Custom Table

HTML

```
<canvas-area xmlns="http://www.w3.org/1999/xhtml" canvas-uuid="a4c909fb-70d1-44f1-a95a-17a0eada6bf6">
	<custom-html publish-uuid="eff0b9bd-4009-4d83-ade8-46c40587ed97" width="880" height="450" top="130" left="10" rotation="0" name="custom_table" on-click="none" style="z-index: 2;"></custom-html>
	<report-output publish-uuid="90a601cc-a148-40d8-a460-15daab8a238b" report-uuid="3e842fae-02f7-4ad3-a632-ca267e0078da" height="125" top="4" left="8" name="Campaign Summary" width="880" display-type="CANVAS" on-click="none" style="z-index: 1;"></report-output>
</canvas-area>
```

From Yellowfin 9.5, publish-uuid replaces the older widget-uuid. This page has been updated to reflect the updated UUID. For Yellowfin users on older versions, we recommend you upgrade, or use the name attribute in place of publish-uuid in the interim.

JS

```
this.outputTypes = []; //Create an array of outputKeys and what report API they came from for easy removal

this.onRender = function () {
	let campaignSummary = this.apis.canvas.select('Campaign Summary');
	campaignSummary.onReportLoad.then(() => {
		let reportAPI = campaignSummary.reportAPI;
		let outputKey = reportAPI.registerOutputType('dataset', reportDataset => {
			let tableData = [];
			let tableElement = document.createElement('table');

			//Give the table a class name so we can style it
			tableElement.className = "codeModeExampleTable";

			let tHead = document.createElement('thead');
			let headerRow = document.createElement('tr');
			tHead.appendChild(headerRow);
			tableElement.appendChild(tHead);

			//Loop over all the reports fields to get their names. The fields will be
			//in the same order as the columns in the dataset
			reportAPI.fields.forEach(field => {
				let th = document.createElement('th');
				th.innerText = field.name;
				headerRow.appendChild(th);
			});

			let tbody = document.createElement('tbody');
			tableElement.appendChild(tbody);
			//Loop over the rows in the dataset and create a table row at each level.
			reportDataset.forEach(rowData => {
				let tr = document.createElement('tr');

				//Loop over the columns and append the formattedValue from each item
				rowData.forEach(columnData => {
					let td = document.createElement('td');
					//columnData will have rawValue, formattedValue and htmlFormattedValue.
					td.innerText = columnData.formattedValue;
					tr.appendChild(td);
				});
				tbody.appendChild(tr);
			  });

			  //Now we've done all that we want to get the custom_table element and insert the table there
			  let customTable = this.apis.canvas.select('custom_table');
			  customTable.innerHTML = ''; //Clear any previous table that might be there
			  customTable.appendChild(tableElement); //Append our new table

		});

		this.outputTypes.push({
			key: outputKey,
			reportAPI: reportAPI
		});

	});
};

this.onRemove = function () {
	//Remove the output types so that we aren't generating extra tables each time we come back to this
	//tab
	this.outputTypes.forEach(outputObject => {
		outputObject.reportAPI.removeOutputType(outputObject.key)
	});
};
```

CSS

```
table.codeModeExampleTable {
	width:100%;
	height:100%;
	border-collapse:separate;
	border-spacing:0px;
}

table.codeModeExampleTable thead {
	background-color: #009EEC;
}

table.codeModeExampleTable thead th {
	border-left: 1px solid white;
}

table.codeModeExampleTable tbody td {
	border:1px solid #ccc;
	border-top:none;
}
```