



Lab 02C

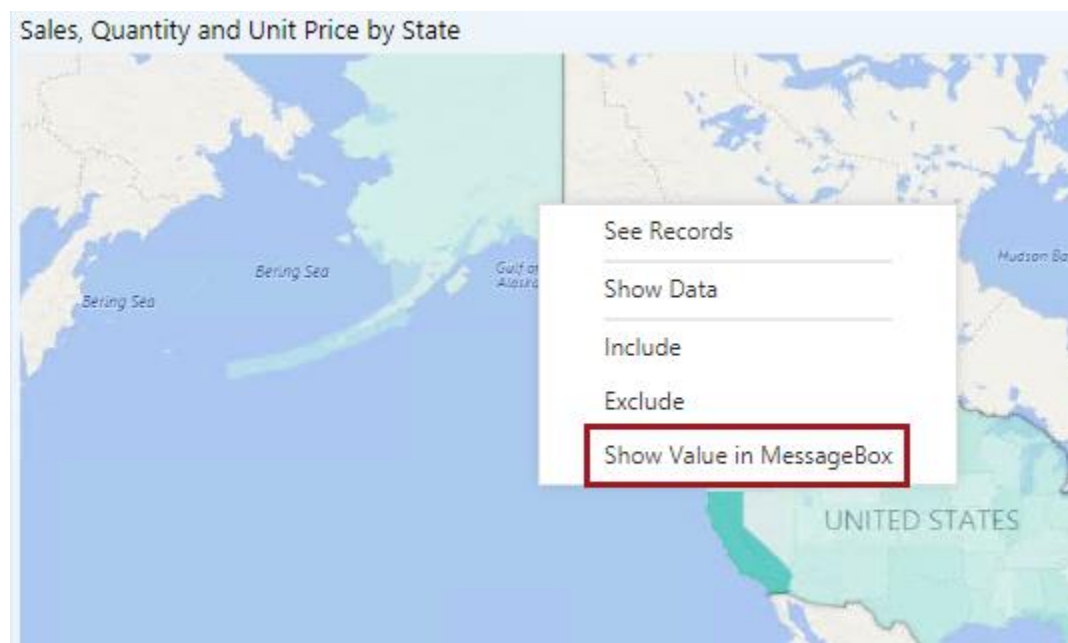
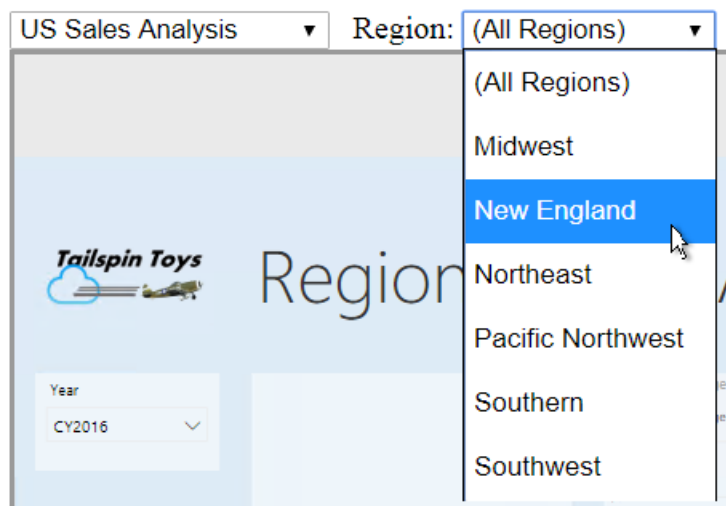
Exploring the Power BI JavaScript API

Overview

The estimated time to complete the lab is 30 minutes

*You must have completed **Lab 02B** before commencing this lab.*

In this lab, you will explore the sample web application and many Power BI JavaScript API operations. You will then add client-side filtering to the **EmbedReport.aspx** web form to enable filtering by region, and also a context menu.



Exploring the Sample Web Application


In this exercise, you will explore the **Report Embed Sample** web application to learn about supported capabilities of the JavaScript API.

Outputting Embed Values

In this task, you will modify the **EmbedReport.aspx** web form logic to output specific embed variable values.

1. In Visual Studio, in the **PowerBIEmbedding** project, open the **EmbedReport.aspx** item.
2. To output variable values, beneath the **asp:DropDownList** element, add the following HTML elements:

For convenience, the elements can be copied from the <CourseFolder>PowerBIDevIAD\Lab02C\Assets\Snippets.txt file.

```
12      <asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True" OnSelected
13      <br />
14      
15      <div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px;" />
```

ASP.NET

```
<span>Embed Token: </span>
<textarea cols="120" rows="2" readonly="readonly"><% =this.embedToken %></textarea><br />
<span>Embed URL: </span>
<textarea cols="120" rows="2" readonly="readonly"><% =this.embedUrl %></textarea><br />
<span>Report Id: </span>
<textarea cols="120" rows="2" readonly="readonly"><% =this.reportId %></textarea><br />
```

3. Start the application.
4. When the web session opens, navigate to the **Embed Reports** page.
5. Leave the web session open.

You will copy values from this page in the next task.

Exploring the Sample Web Application

In this task, you will explore the **Report Embed Sample** web application.

1. Within the existing Google Chrome window, add a new session (tab), and then navigate to <https://microsoft.github.io/PowerBI-JavaScript/demo/v2-demo/index.html>.

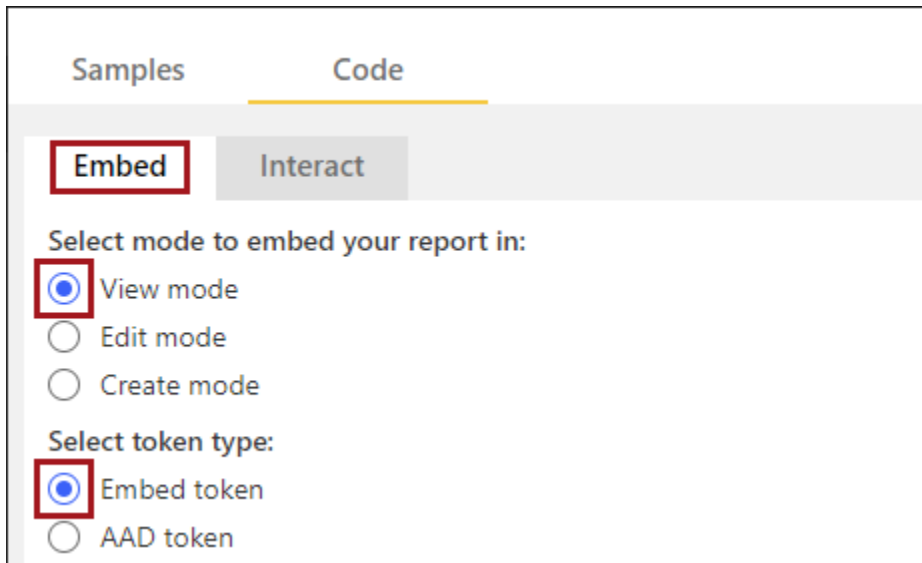
For convenience, the URL can be copied from the <CourseFolder>\PowerBIDevIAD\Lab02C\Assets\Snippets.txt file.

The Microsoft Power BI Embedded Playground sample web application was designed to enable experimenting with Power BI embedded analytics without developing an application. It works with either a sample content (report, report visual, dashboard, tile or Q&A), or with your own content. It also allows creating and editing reports. In this lab, you will embed Power BI content from your app workspace.

2. Select the **Code** page.



3. In the **Embed** tab, notice that the default report mode is **View Mode**, and the default token type is **Embed Token**—there is no need to change these.



- Copy all embed values from your web app into the three boxes in the sample web application.

*When copying the values, be sure to select all text in the boxes (press **Ctrl+A**) before pasting.*

US Sales Analysis ▼

Embed Token:

Embed URL:

Report Id:

Fill in the fields below to get the code to embed your report.

Embed Token

Embed URL

Report Id

- In the **Code** box, review the JavaScript code designed to use these values to embed the report into the lower pane.

```
Code
```

▶ Run Copy

```
// Read embed application token from textbox
var txtAccessToken = $('#txtAccessToken').val();

// Read embed URL from textbox
var txtEmbedUrl = $('#txtReportEmbed').val();
```

- To run the code to embed the report, in the **Code** box, at the top-left corner, click **Run**.

```
Code
```

▶ Run Copy

- In the **Embedded View** pane (below), ensure that your report loads.

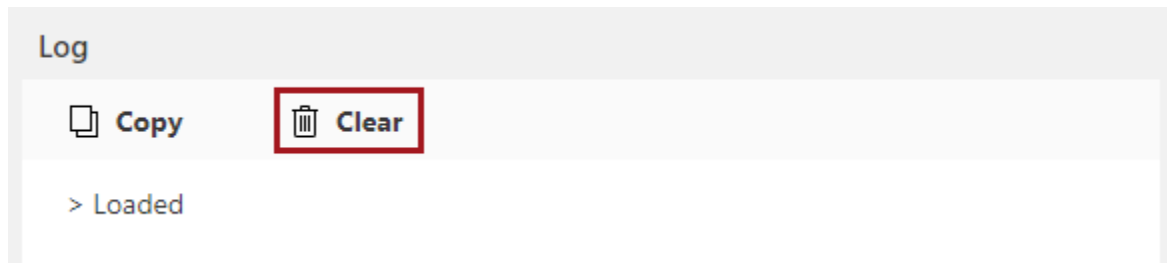
*This will be reported by feedback loaded in the **Log** box.*

```
Log
```

Copy Clear

> Loaded

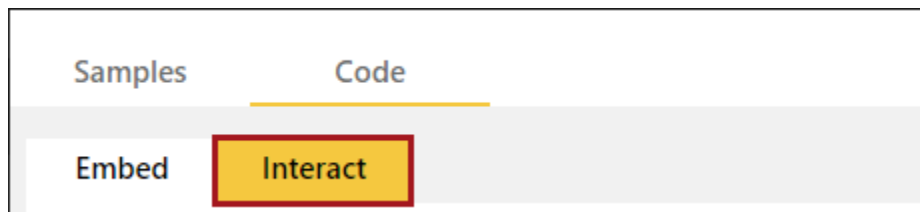
8. To clear the log messages, in the **Log** box, click **Clear**.



Exploring the JavaScript API Capabilities

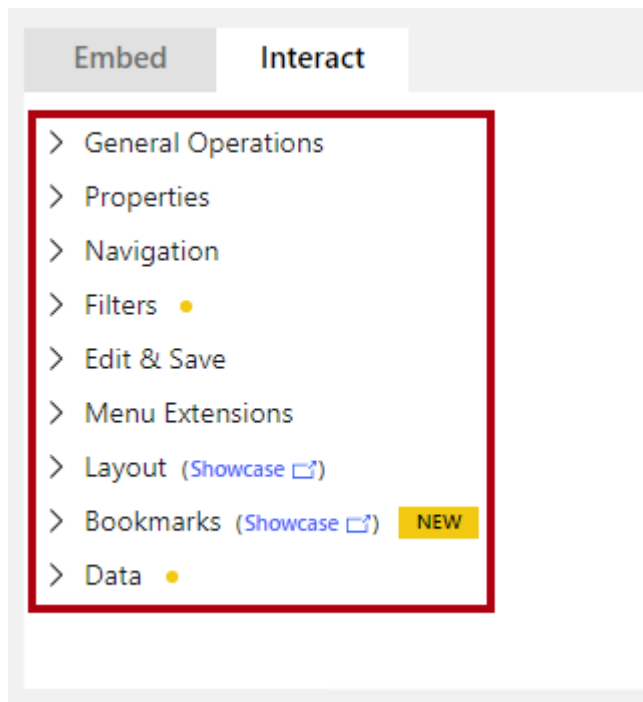
In this task, you will explore several capabilities supported by the JavaScript API.

1. Select the **Interact** tab.

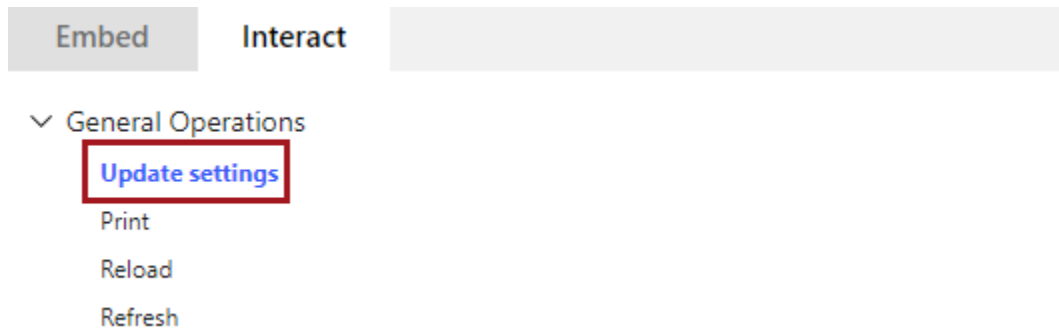


It is important to understand that all the programmatic operations you work with in this task are achieved by client-side script.

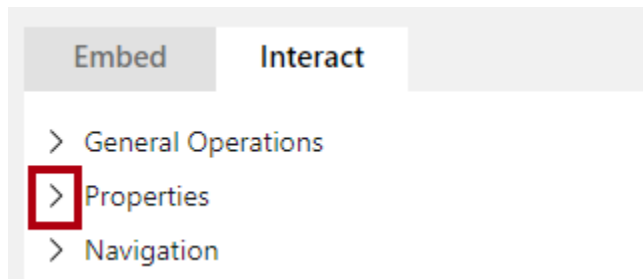
2. Notice the list of categories, and that each can be expanded.



- Expand the **General Operations** category, and then select the **Update Settings** operation.



- In the **Code** box, review the JavaScript code designed to perform the selected operation, and then click **Run**.
- Notice that the report **Filters** pane (located at the very right) was removed.
- Run the following operations:
 - Reload—notice that the **Filters** pane returns
 - Refresh (refresh retrieves the latest data from the Power BI service; reload retrieves the latest report definition from the service)
 - Full screen—press **Escape** to exist full screen mode
- Clear the log messages.
- Collapse the **General Operations** category, and then expand the **Properties** category.



- Run each of the operations, reviewing the code, and output in the **Log** box.
- Clear the **Log** box.
- Collapse the **Properties** category.

12. Run each of the following operations, reviewing the code and output, and clearing the **Log** box between runs.

Category	Operations	Directions
Navigation	Page – Set active	Notice that the second page is in focus
	Page changed event	Once running, use the report page tabs to make the first report page active, and review the output messages
Filters	Get report filters	Review the output describing current filter context. Open the Filters pane, apply a different filter, and run again.
	Remove report filters	Notice that the report filters have been removed. They will be restored when the report is next reloaded.
Menu Extensions	Extend options menu	Click the ellipsis at the top-right of the Quantity card, and then select the new Extend Options Menu item. Review the output JSON document describing the report, page, visual and selected command.
	Extend menu content	In the Sales and Avg Price by Month visual, right-click any month column, and then select Extend Context Menu . Review the output JSON document describing the report, page, visual and data point details. <i>You will add a context menu to your app in the next exercise.</i>
Layout	Apply custom layout	Notice that the report size increases. Reload the report (in the General Operations category).
Bookmarks	Enable bookmarks pane	Notice that the Bookmarks pane opens Interact with the report by selecting different bookmarks.
	Play bookmarks	Scroll down, and notice that the bookmark navigation bar along the bottom of the report. Use the forward arrow (located at the very right), to navigate forward through the bookmarks.
	Exit play bookmarks mode	Notice that the bookmark navigation bar has been removed
	Disable bookmarks pane	Notice that the Bookmarks pane is closed
Data	Data selected event	Run, click on a column in the report, and then review the log output.
	Export visual data summarized	Run, and review the log output describing the data for the VisualContainer3 visual (Category slicer).

13. Close the sample web application session (tab).

14. Close the **PowerBIEmbedding** web session, and then stop debugging.

Removing the Embed Values Output

In this task, you will remove the embed values output from the web form.

1. In the **EmbedReport.aspx** item, remove the HTML elements added in the first task of this exercise.

```
11 <form id="form1" runat="server">
12   <asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True" OnSelectedIndexChanged
13   <br />
14   <span>Embed Token: </span>
15   <textarea cols="120" rows="2" readonly="readonly"><% =this.embedToken %></textarea><br />
16   <span>Embed URL: </span>
17   <textarea cols="120" rows="2" readonly="readonly"><% =this.embedUrl %></textarea><br />
18   <span>Report Id: </span>
19   <textarea cols="120" rows="2" readonly="readonly"><% =this.reportId %></textarea><br />
20   <div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px; />
```

2. Save the **EmbedReport.aspx** item.

Adding Client-Side Filtering


In this exercise, you will add functionality to the **EmbedReport.aspx** web form by using the Power BI JavaScript API to filter the embedded report by all regions, or a specific region.

Adding a Filter Dropdown List

In this task, you will modify the **EmbedReport.aspx** web form by adding a filter dropdown list.

1. To add a dropdown list, beneath the **
** element after the **asp:DropDownList** element, add the following HTML elements:

*For convenience, the elements can be copied from the **<CourseFolder>\PowerBIDev\AD\Lab02C\Assets\Snippets.txt** file.*

```
9 <body>
10 <script src="/scripts/powerbi.js"></script>
11 <form id="form1" runat="server">
12 <asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True" OnSelected:
13 <br />
14 
15 <div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px;" />
```

HTML


```
<span>Region: </span>
<select id="ddlFilterRegion" onchange="filterRegion(); return false;">
  <option selected="selected" value="">(All Regions)</option>
  <option value="Midwest">Midwest</option>
  <option value="New England">New England</option>
  <option value="Northeast">Northeast</option>
  <option value="Pacific Northwest">Pacific Northwest</option>
  <option value="Southern">Southern</option>
  <option value="Southwest">Southwest</option>
</select>
```

The dropdown list presents each region, with the first item representing all regions. The **onchange** event handler will invoke the **filterRegion** function that you will add to the **script** element in the next task.

Adding Client-Side Filter Logic

In this task, you will add the **filterRegion** function that will perform client-side filter logic.

1. To add the **filterRegion** function, inside the **script** element, add the following JavaScript function:

```
53      // Embed the report within the div element
54      var report = powerbi.embed(embedDiv, config);
55
56      
57      </script>
58  </form>
59 </body>
60 </html>
```

JavaScript

```
function filterRegion() {

    var report = powerbi.embeds[0];
    var ddl = document.getElementById("ddlFilterRegion");
    var region = ddl.options[ddl.selectedIndex].value;

    if (region == "*") {
        report.removeFilters()
            .catch(error => { console.log(error); });
        return;
    }

    const basicFilter = {
        "$schema": "http://powerbi.com/product/schema#basic",
        "target": {
            "table": "Region",
            "column": "Region"
        },
        "operator": "In",
        "values": [
            region
        ]
    }

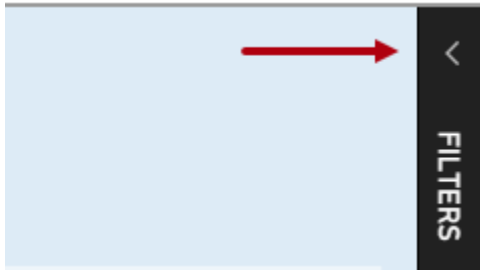
    report.setFilters([basicFilter])
        .catch(error => { console.log(error); });
}
```

*The function removes the region filter if **(All Regions)** is selected, otherwise it filters by the selected region.*

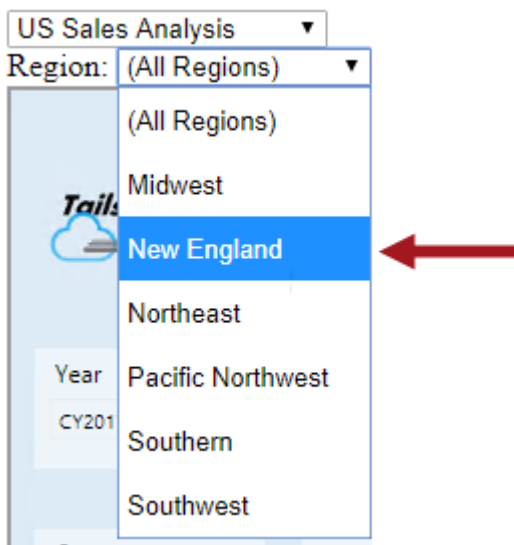
Reviewing the Filter Functionality

In this task, you will review the filter functionality.

1. Start the web application, and then navigate to the **Embed Reports** page.
2. In the **US Sales Analysis** report, open the **Filters** pane.



3. In the **Region** dropdown list, select any **Region**.



4. Notice that the **Region** filter selection has updated.

When setting filters, all items must be submitted as an array, and will replace any existing filters.

5. Notice that the **Sales % All Regions** card displays a value less than 100%, representing the contribution the selected region made over all regions.



6. In the **Region** dropdown list, select any **(All Regions)**.
7. Notice that the report filter has been removed.
8. Notice that the **Sales % All Regions** card displays a value of 100%, as expected.



9. Close the web session, and then stop debugging.

Removing the Client-Side Filter Logic

In this task, you will remove the client-side filter logic from the web form.

1. In the **EmbedReport.aspx** item, remove the HTML elements added in the first task of this exercise.

```

12      <asp:DropDownList ID="ddlReport" runat="server" AutoPostBack="True" OnSelecte
13      <br />
14      <span>Region: </span>
15      <select id="ddlFilterRegion" onchange="filterRegion(); return false;">
16          <option selected="selected" value="*">(All Regions)</option>
17          <option value="Midwest">Midwest</option>
18          <option value="New England">New England</option>
19          <option value="Northeast">Northeast</option>
20          <option value="Pacific Northwest">Pacific Northwest</option>
21          <option value="Southern">Southern</option>
22          <option value="Southwest">Southwest</option>
23      </select>
24      <br />
25      <div id="embedDiv" style="height: 600px; width: 100%; max-width: 1000px;" />

```

2. In the **script** element, remove the **filterRegion** function.

```
54     var report = powerbi.embed(embedDiv, config);
55
56     function filterRegion() {
57
58         var report = powerbi.embeds[0];
59         var ddl = document.getElementById("ddlFilterRegion");
60         var region = ddl.options[ddl.selectedIndex].value;
61
62         if (region == "*") {
63             report.removeFilters()
64                 .catch(error => { console.log(error); });
65             return;
66         }
67
68         const basicFilter = {
69             "$schema": "http://powerbi.com/product/schema#basic",
70             "target": {
71                 "table": "Region",
72                 "column": "Region"
73             },
74             "operator": "In",
75             "values": [
76                 region
77             ]
78         }
79
80         report.setFilters([basicFilter])
81             .catch(error => { console.log(error); });
82     }
83 </script>
```



3. Save the **EmbedReport.aspx** item.

Adding a Context Menu


In this exercise, you will add functionality to the **EmbedReport.aspx** web form by using the Power BI JavaScript API to extend visuals with a context menu option.

Adding a Context Menu

In this task, you will modify the **EmbedReport.aspx** web form by adding a context menu.

1. In the **EmbedReport.aspx** item, in the **config** variable assignment, add a comma at the end of the line for the **navContentPaneEnabled** setting.


```
30  var config = {  
31      type: 'report',  
32      tokenType: models.TokenType.Embed,  
33      accessToken: embedToken,  
34      embedUrl: embedUrl,  
35      id: reportId,  
36      settings: {  
37          filterPaneEnabled: true,  
38          navContentPaneEnabled: true,  
39      }  
40  };
```



2. Inside the **settings** parameter, add the **extensions** setting.

For convenience, the code can be copied from the
<CourseFolder>\PowerBIDev\AD\Lab02C\Assets\Snippets.txt file.

```
30  var config = {  
31      type: 'report',  
32      tokenType: models.TokenType.Embed,  
33      accessToken: embedToken,  
34      embedUrl: embedUrl,  
35      id: reportId,  
36      settings: {  
37          filterPaneEnabled: true,  
38          navContentPaneEnabled: true,  
39      }  
40  };  
41
```



JavaScript

```
extensions: [  
  {  
    command: {  
      name: "cmdShowValue",  
      title: "Show Value in MessageBox",  
      extend: {  
        visualContextMenu: {  
          title: "Show Value in MessageBox"  
        }  
      }  
    }  
  }  
]
```

```
30  var config = {  
31    type: 'report',  
32    tokenType: models.TokenType.Embed,  
33    accessToken: embedToken,  
34    embedUrl: embedUrl,  
35    id: reportId,  
36    settings: {  
37      filterPaneEnabled: true,  
38      navContentPaneEnabled: true,  
39      extensions: [  
40        {  
41          command: {  
42            name: "cmdShowValue",  
43            title: "Show Value in MessageBox",  
44            extend: {  
45              visualContextMenu: {  
46                title: "Show Value in MessageBox"  
47              }  
48            }  
49          }  
50        }  
51      ]  
52    }  
53  }  
};
```

This setting adds a context menu item with the title "Show Value in Message Box".

3. Inside the **script** element, add the following JavaScript event handler:

```
54 |
55 |         // Embed the report within the div element
56 |         var report = powerbi.embed(embedDiv, config);
57 |
58 |         </script>
59 |     </form>
60 |
61 | </body>
62 | </html>
```

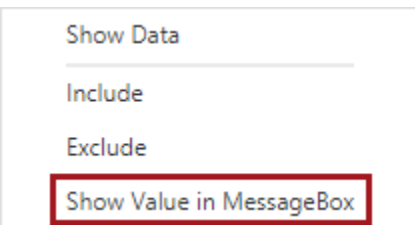
JavaScript

```
// Add an event handler for the commandTriggered event
report.on("commandTriggered", function (command) {
    // Determine the command detail
    var commandDetails = command.detail;

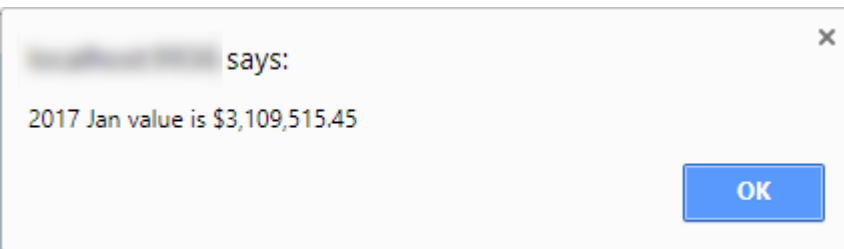
    // If the command is cmdShowValue, show a message box
    if (commandDetails.command == "cmdShowValue") {
        // Retrieve specific details from the selected data point
        var category = commandDetails.dataPoints[0].identity[0].equals;
        var value = commandDetails.dataPoints[0].values[0].formattedValue;

        // Open message box
        alert(category + " value is " + value);
    }
});
```

4. Start the web application, and then navigate to the **Embed Reports** page.
5. In the column chart, right-click any month column, and then select **Show Value in Message Box**.



6. Verify that the browser opens a message box describing the selected data element.



7. To dismiss the message box, click **OK**.

8. Test the context menu on a bar chart bar, and also a filled map state (**Geographic Analysis** page).
9. Close the web browser, and stop debugging.
10. Close all open project items.

*You will modify the solution to apply row-level security (RLS) to enforce data permissions in **Lab 02C**.*

Summary

In this lab, you explored the sample web application and many Power BI JavaScript API operations. You then added client-side filtering to the **EmbedReport.aspx** web form to enable filtering by region.

Terms of Use

© 2017-2018 Microsoft. All rights reserved.

By using this hands-on lab, you agree to the following terms:

The technology/functionality described in this hands-on lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the hands-on lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. Without written permission, you may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this hands-on lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION WITHOUT WRITTEN PERMISSION IS EXPRESSLY PROHIBITED.

THIS HANDS-ON LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCTIONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK If you give feedback about the technology features, functionality and/or concepts described in this hands-on lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER This lab contains only a portion of new features and enhancements in Microsoft Power BI. Some of the features might change in future releases of the product.

Document Version

#	Date	Author	Comments
1	22-JUL-2017	Peter Myers	Visual Studio 2017 v15.2 (26430.14) Power BI service v13.0.1930.145
2	31-DEC-2017	Peter Myers	Visual Studio 2017 v15.2 (26430.14) Power BI service v13.0.3495.220 Microsoft.IdentityModel.Clients.ActiveDirectory v3.17.3 Microsoft.PowerBI.Api v2.0.8 MicrosoftPowerBI.JavaScript v2.4.2 Microsoft.Rest.ClientRuntime v2.0.1 Newtonsoft.Json v7.01
3	21-FEB-2018	Peter Myers	Visual Studio 2017 v15.2 (26430.14) Power BI service v13.0.4385.126 Microsoft.IdentityModel.Clients.ActiveDirectory v3.19.1 Microsoft.PowerBI.Api v2.0.10 MicrosoftPowerBI.JavaScript v2.4.7 Microsoft.Rest.ClientRuntime v2.3.10 Newtonsoft.Json v11.0.1
4	21-APR-2018	Peter Myers	(Updated lab steps for revised web sample app only.)
5	30-JUN-2018	Peter Myers	(Resequenced from Lab 02B to Lab 02C) Power BI service v13.0.5702.156 Microsoft.IdentityModel.Clients.ActiveDirectory v3.19.8 Microsoft.PowerBI.Api v2.0.12 MicrosoftPowerBI.JavaScript v2.5.1 Microsoft.Rest.ClientRuntime v2.3.11 Newtonsoft.Json v11.0.2