Microsoft Power BI

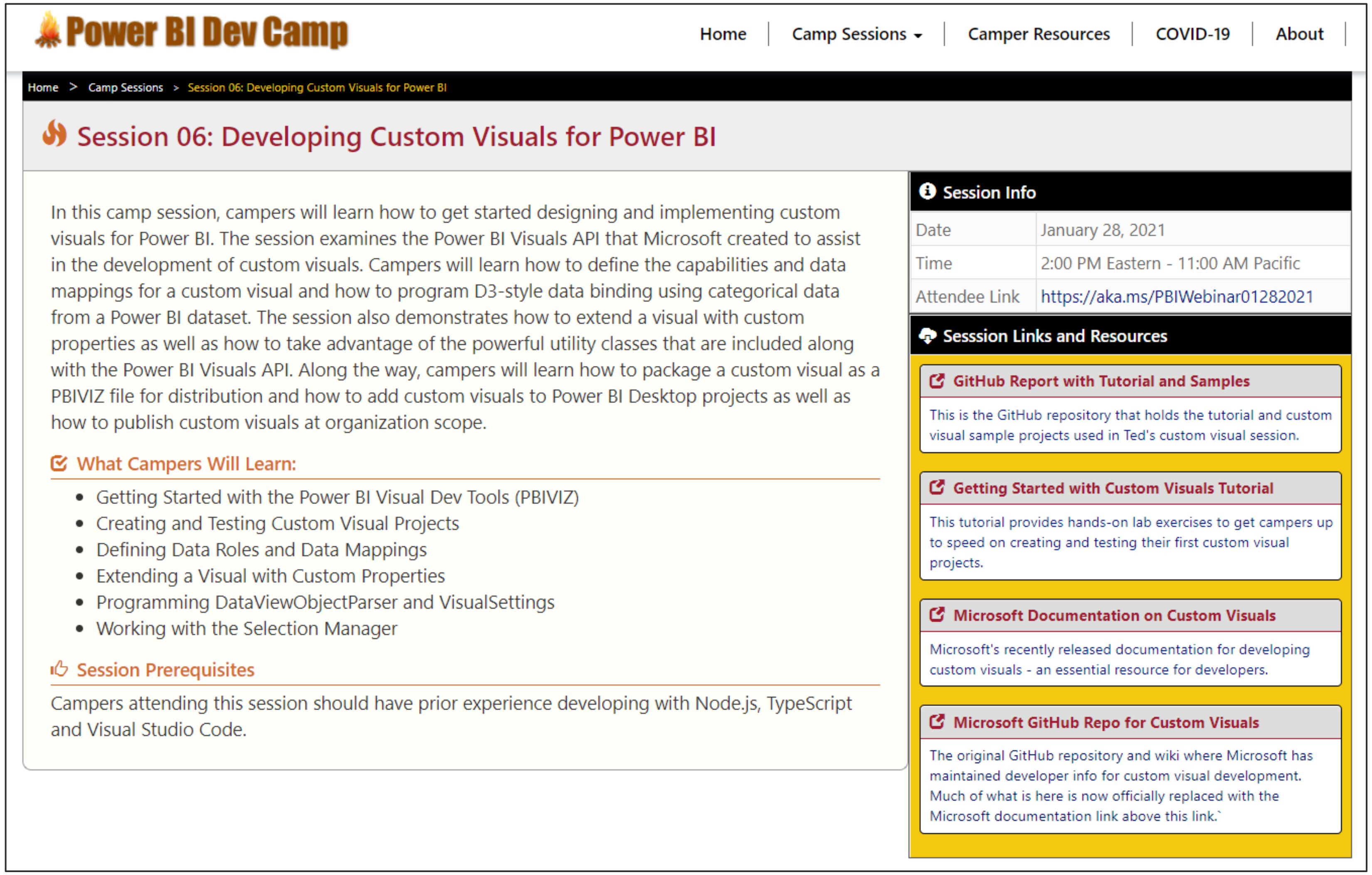
Power BI Dev Camp – Session 6

Developing Custom Visuals for Power BI

Ted Pattison
Principal Program Manager
Customer Advisory Team (CAT) at Microsoft

Welcome to Power BI Dev Camp

- Power BI Dev Camp Portal - https://powerbidevcamp.net



Power BI Dev Camp

Home | Camp Sessions | Camper Resources | COVID-19 | About

Home > Camp Sessions > Session 06: Developing Custom Visuals for Power BI

Session 06: Developing Custom Visuals for Power BI

In this camp session, campers will learn how to get started designing and implementing custom visuals for Power BI. The session examines the Power BI Visuals API that Microsoft created to assist in the development of custom visuals. Campers will learn how to define the capabilities and data mappings for a custom visual and how to program D3-style data binding using categorical data from a Power BI dataset. The session also demonstrates how to extend a visual with custom properties as well as how to take advantage of the powerful utility classes that are included along with the Power BI Visuals API. Along the way, campers will learn how to package a custom visual as a PBIVIZ file for distribution and how to add custom visuals to Power BI Desktop projects as well as how to publish custom visuals at organization scope.

What Campers Will Learn:

- Getting Started with the Power BI Visual Dev Tools (PBIVIZ)
- Creating and Testing Custom Visual Projects
- Defining Data Roles and Data Mappings
- Extending a Visual with Custom Properties
- Programming DataViewObjectParser and VisualSettings
- Working with the Selection Manager

Session Prerequisites

Campers attending this session should have prior experience developing with Node.js, TypeScript and Visual Studio Code.

Session Info

Date	January 28, 2021
Time	2:00 PM Eastern - 11:00 AM Pacific
Attendee Link	https://aka.ms/PBIWebinar01282021

Sesssion Links and Resources

GitHub Report with Tutorial and Samples

This is the GitHub repository that holds the tutorial and custom visual sample projects used in Ted's custom visual session.

Getting Started with Custom Visuals Tutorial

This tutorial provides hands-on lab exercises to get campers up to speed on creating and testing their first custom visual projects.

Microsoft Documentation on Custom Visuals

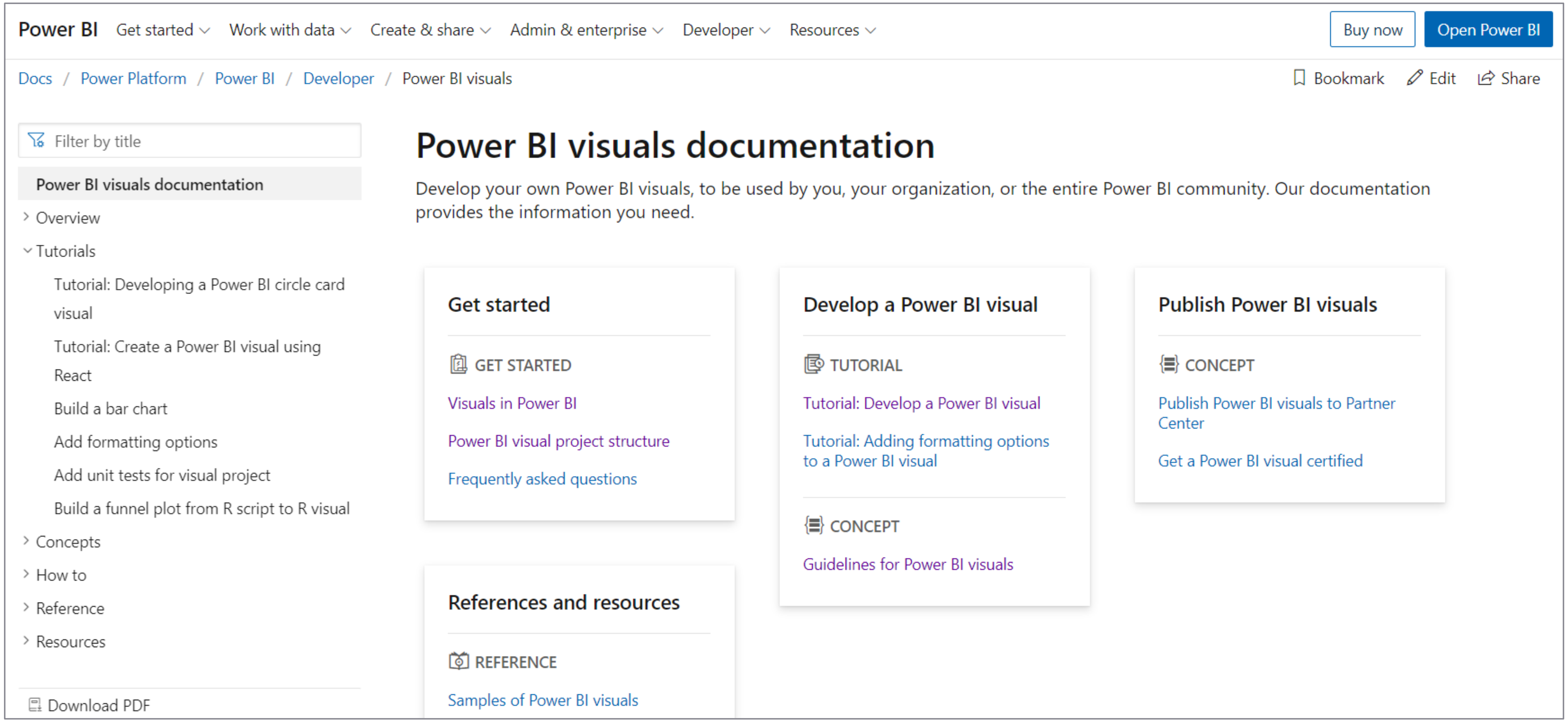
Microsoft's recently released documentation for developing custom visuals - an essential resource for developers.

Microsoft GitHub Repo for Custom Visuals

The original GitHub repository and wiki where Microsoft has maintained developer info for custom visual development. Much of what is here is now officially replaced with the Microsoft documentation link above this link.`

Microsoft Custom Visual Documentation

- https://docs.microsoft.com/en-us/power-bi/developer/visuals/



Power BI visuals documentation
Overview
Tutorials
Tutorial: Developing a Power BI circle card visual
Tutorial: Create a Power BI visual using React
Build a bar chart
Add formatting options
Add unit tests for visual project
Build a funnel plot from R script to R visual
Concepts
How to
Reference
Resources
Download PDF

Power BI visuals documentation

Develop your own Power BI visuals, to be used by you, your organization, or the entire Power BI community. Our documentation provides the information you need.

Get started

GET STARTED
Visuals in Power BI
Power BI visual project structure
Frequently asked questions

Develop a Power BI visual

TUTORIAL
Tutorial: Develop a Power BI visual
Tutorial: Adding formatting options to a Power BI visual

CONCEPT
Guidelines for Power BI visuals

Publish Power BI visuals

CONCEPT
Publish Power BI visuals to Partner Center
Get a Power BI visual certified

References and resources

REFERENCE
Samples of Power BI visuals

Agenda

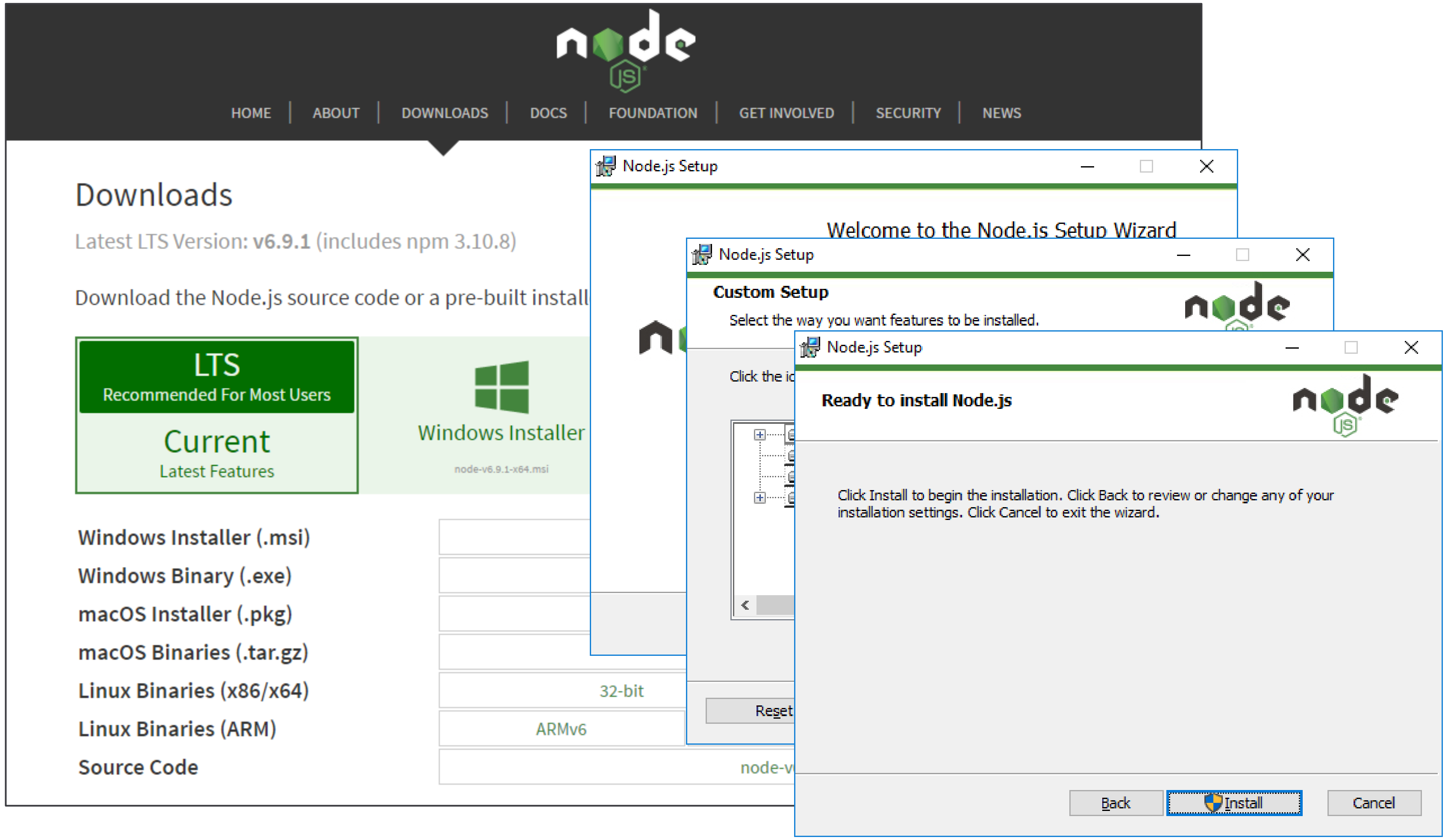
- Installing the Power BI Developer Tools
- Creating Your First Custom Visual
- Defining Data Roles and Data Mappings
- Extending a Visual with Custom Properties
- Implementing Highlighting with SelectionManager
- Custom Visual Packaging and Distribution

Installing the Power BI Developer Toolchain

- **Install Node.JS**
  - Installs Node Package Manage (npm)
- **Install Visual Studio Code**
  - Lightweight Alternative to Visual Studio for Node.js Development
- **Install the Power BI Developer Tools (pbiviz)**
  - Install using Node Package Manager (npm)
- **Create and install a local self-signed certificate**
  - Install using Power BI visuals CLI tool (pbiviz)

Installing node.js

- https://nodejs.org/en/download/



Downloads

Latest LTS Version: **v6.9.1** (includes npm 3.10.8)

Download the Node.js source code or a pre-built install

LTS
Recommended For Most Users

Current
Latest Features

Windows Installer
node-v6.9.1-x64.msi

Windows Installer (.msi)	
Windows Binary (.exe)	
macOS Installer (.pkg)	
macOS Binaries (.tar.gz)	
Linux Binaries (x86/x64)	32-bit
Linux Binaries (ARM)	ARMv6
Source Code	node-v

Node.js Setup

Welcome to the Node.js Setup Wizard

Node.js Setup

Custom Setup

Select the way you want features to be installed.

Click the ic

Reset

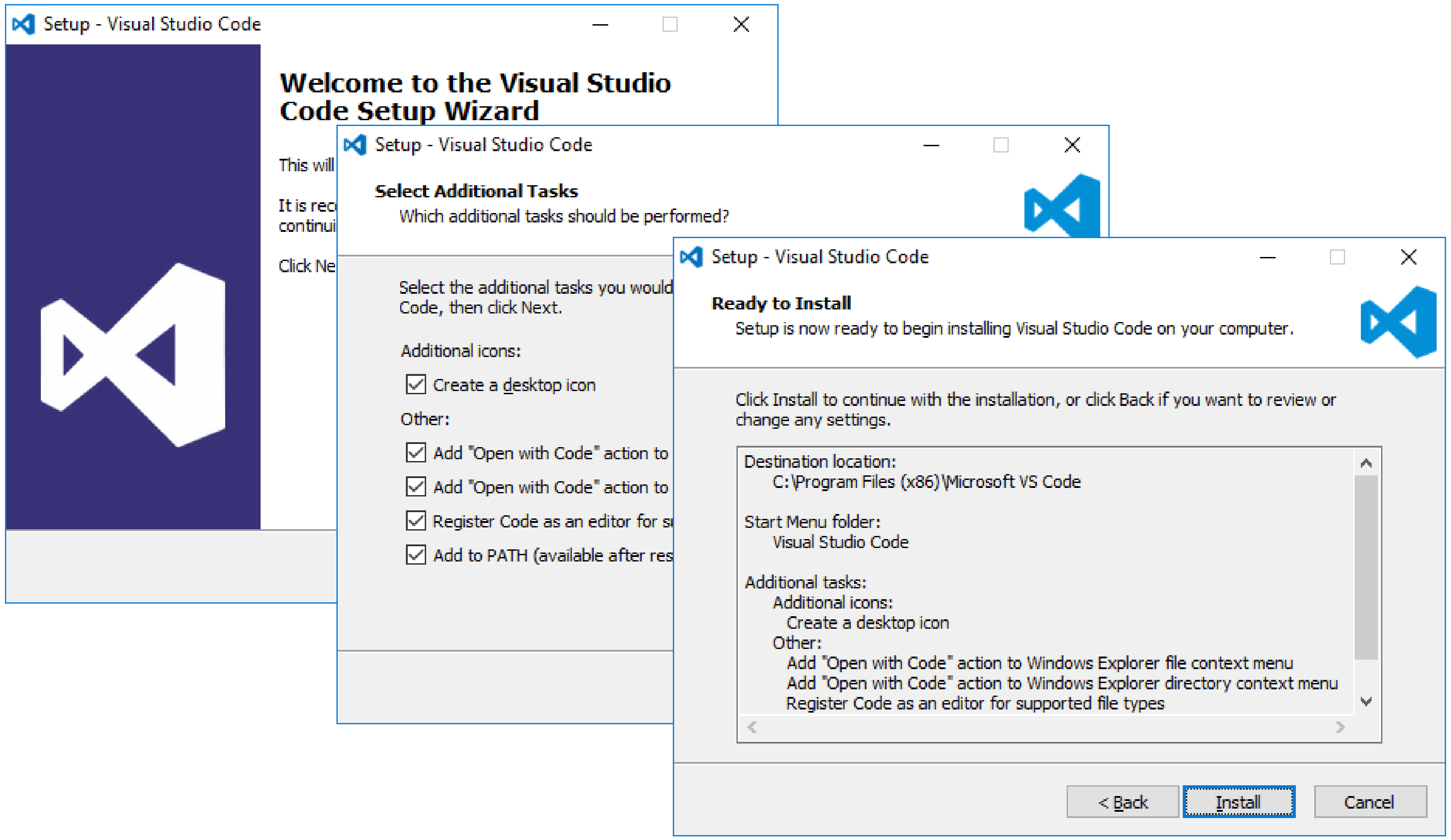
Node.js Setup

Ready to install Node.js

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Install Visual Studio Code

- http://code.visualstudio.com/



Setup - Visual Studio Code

Welcome to the Visual Studio Code Setup Wizard

This will

It is rec
continui

Click Ne

Setup - Visual Studio Code

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would
Code, then click Next.

Additional icons:
☑ Create a desktop icon

Other:
☑ Add "Open with Code" action to
☑ Add "Open with Code" action to
☑ Register Code as an editor for s
☑ Add to PATH (available after res

Setup - Visual Studio Code

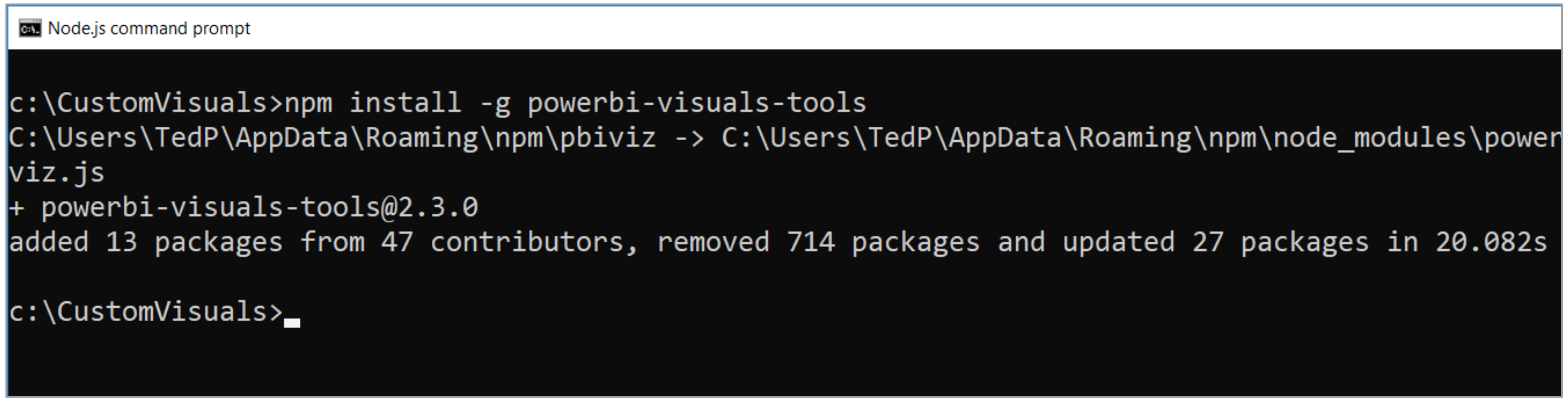
Ready to Install
Setup is now ready to begin installing Visual Studio Code on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
    C:\Program Files (x86)\Microsoft VS Code

Start Menu folder:
    Visual Studio Code

Additional tasks:
    Additional icons:
      Create a desktop icon
    Other:
      Add "Open with Code" action to Windows Explorer file context menu
      Add "Open with Code" action to Windows Explorer directory context menu
      Register Code as an editor for supported file types
```

Power BI Visual CLI Tool (PBIVIZ)

- **What is the Power BI Custom Visual Tool?**
  - Command-line utility for cross-platform dev
  - Use it with Visual Studio or Visual Studio Code
  - Requires that you first install node.js
  - Install by running command from node.js command prompt
    `npm install -g powerbi-visuals-tools`

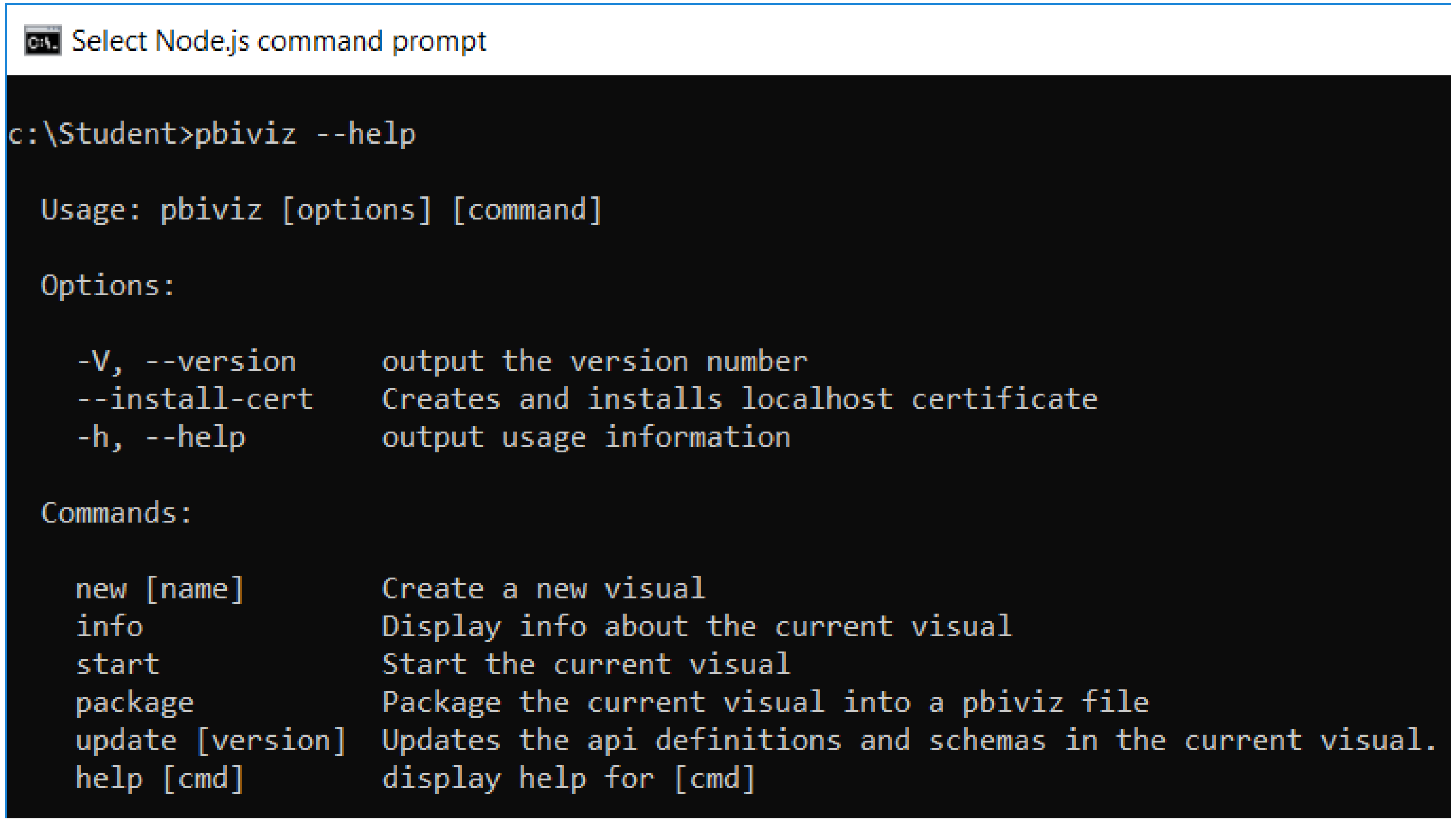


Node.js command prompt

```
c:\CustomVisuals>npm install -g powerbi-visuals-tools
C:\Users\TedP\AppData\Roaming\npm\pbiviz -> C:\Users\TedP\AppData\Roaming\npm\node_modules\power
viz.js
+ powerbi-visuals-tools@2.3.0
added 13 packages from 47 contributors, removed 714 packages and updated 27 packages in 20.082s

c:\CustomVisuals>
```

Getting Started with PBIVIZ

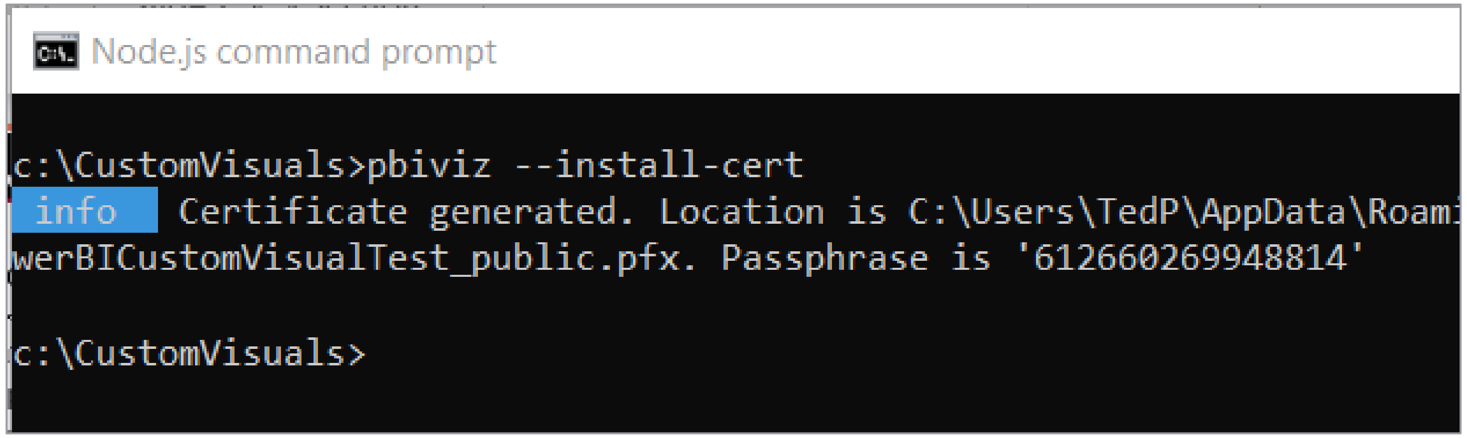
- **PBIVIZ.EXE is a command-line utility**
  - You execute PBIVIZ commands from the NODE.JS command line



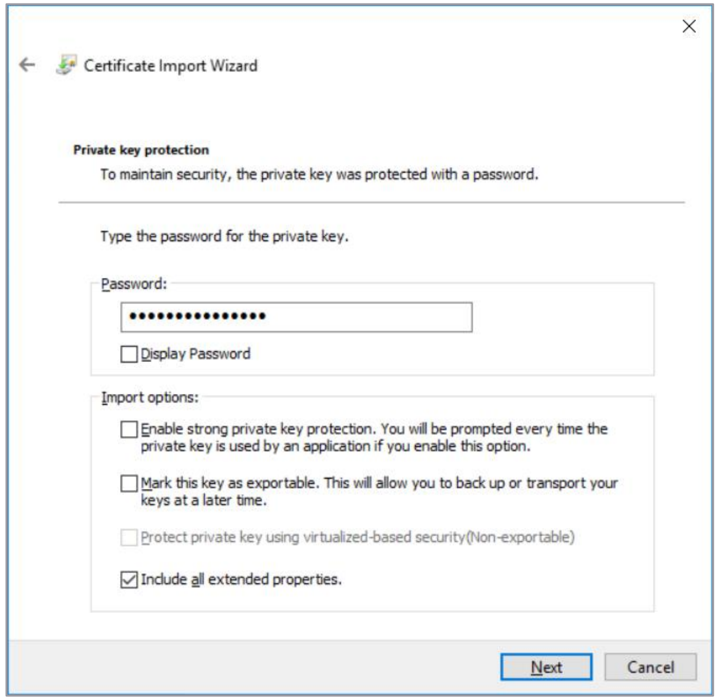
```
c:\Student>pbiviz --help

  Usage: pbiviz [options] [command]

  Options:

    -V, --version     output the version number
    --install-cert    Creates and installs localhost certificate
    -h, --help        output usage information

  Commands:

    new [name]        Create a new visual
    info              Display info about the current visual
    start             Start the current visual
    package           Package the current visual into a pbiviz file
    update [version]  Updates the api definitions and schemas in the current visual.
    help [cmd]        display help for [cmd]
```

Creating a Certificate for Local Testing

- **PBIVIZ provide local web server for testing & debugging**
  - Web server runs locally on developer's workstation in Node.js
  - Makes it possible to test custom visuals in Power BI Service
  - Custom visual resources served up from https://localhost
  - Setup requires creating self-signed SSL certificate
  - SSL certificate created using pbiviz --install-cert command
  - You must copy a passphrase to properly install the certificate


Node.js command prompt

```
c:\CustomVisuals>pbiviz --install-cert
 info   Certificate generated. Location is C:\Users\TedP\AppData\Roami
werBICustomVisualTest_public.pfx. Passphrase is '612660269948814'

c:\CustomVisuals>
```


Certificate Import Wizard

Private key protection

To maintain security, the private key was protected with a password.

Type the password for the private key.

Password: •••••••••••••••

☐ Display Password

Import options:

☐ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

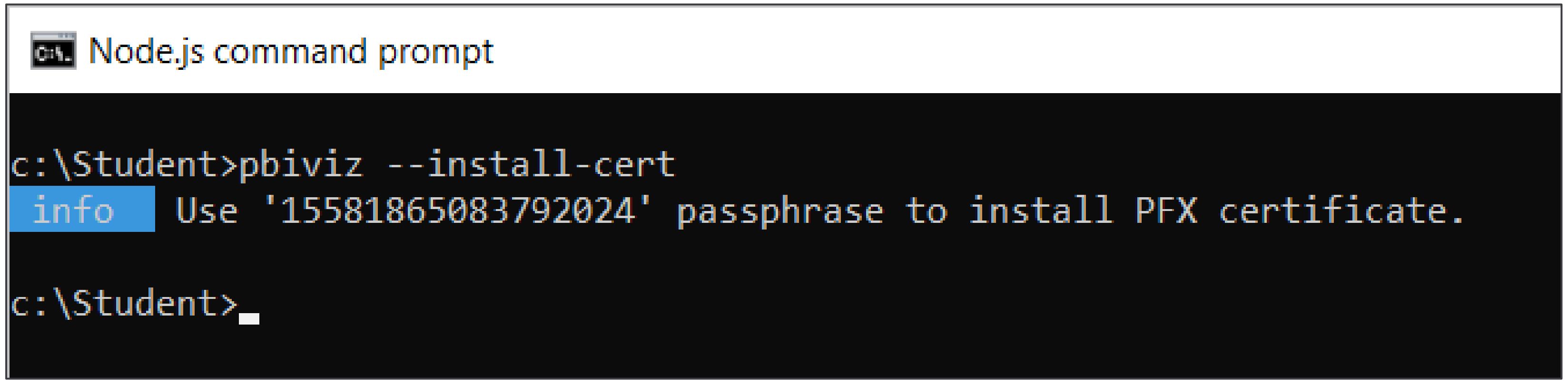
☐ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

☐ Protect private key using virtualized-based security(Non-exportable)

☑ Include all extended properties.

Installing the SSL Certificate

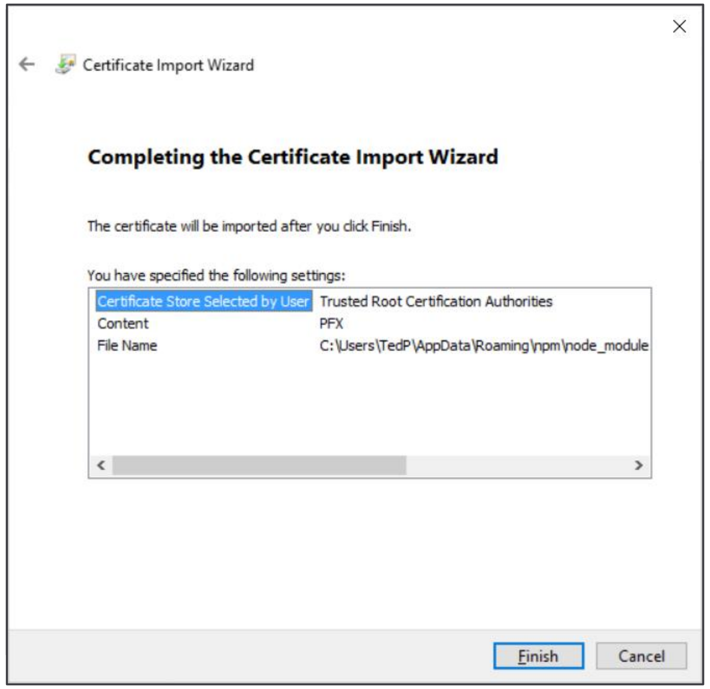
- **Installing certificate enables SSL through https://localhost**
  - Installing certificate is a one time operation – not once per project
  - SSL certificate installed using pbiviz --install-cert command
  - Running --install-cert command starts Certificate Import Wizard



```
c:\Student>pbiviz --install-cert
 info   Use '15581865083792024' passphrase to install PFX certificate.

c:\Student>_
```

The Certificate Import Wizard

- **Wizards steps you through process of installing certificate**
  - You enter certificate passphrase as part of installation process



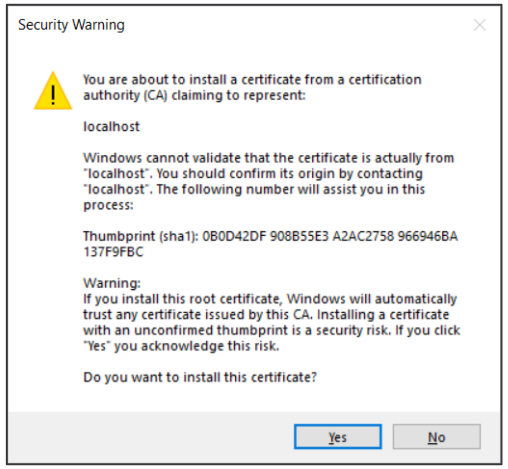
Certificate Import Wizard

Completing the Certificate Import Wizard

The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certification Authorities
Content	PFX
File Name	C:\Users\TedP\AppData\Roaming\npm\node_module



Security Warning

You are about to install a certificate from a certification authority (CA) claiming to represent:

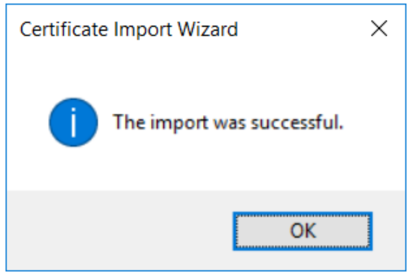
localhost

Windows cannot validate that the certificate is actually from "localhost". You should confirm its origin by contacting "localhost". The following number will assist you in this process:

Thumbprint (sha1): 0B0D42DF 908B55E3 A2AC2758 966946BA 137F9FBC

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?



Certificate Import Wizard

The import was successful.

Agenda

- ✓ Installing the Power BI Developer Tools
- ➢ Creating Your First Custom Visual
- Defining Data Roles and Data Mappings
- Extending a Visual with Custom Properties
- Implementing Highlighting with SelectionManager
- Custom Visual Packaging and Distribution

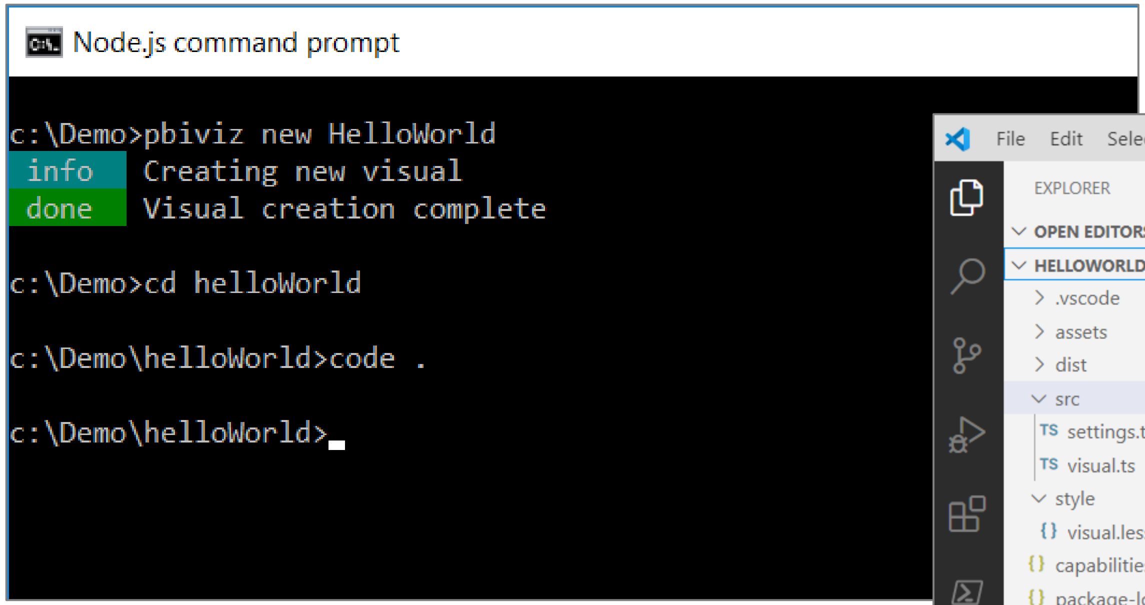
Creating a New Custom Visual Project

- Creating a new project

```
pbiviz new <ProjectName>
```

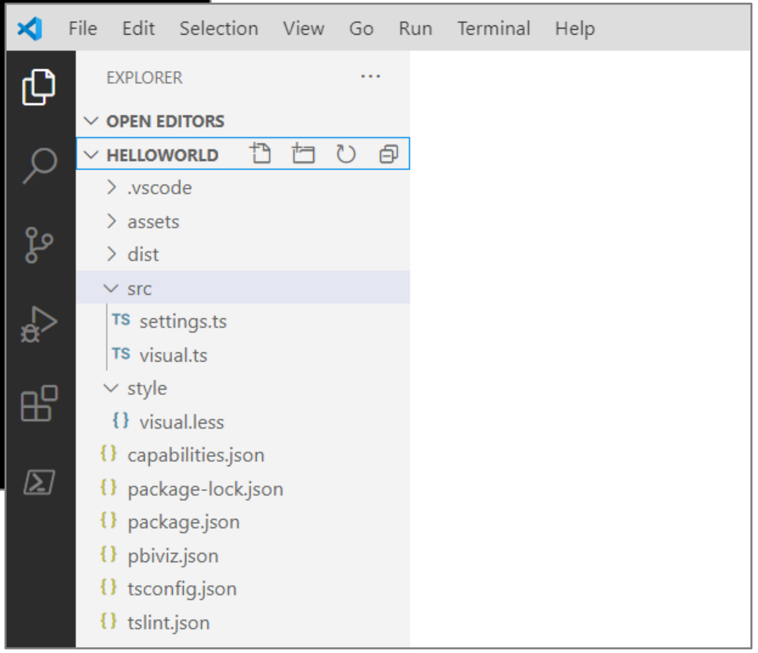
- Open the Project with Visual Studio Code

```
code .
```



Node.js command prompt

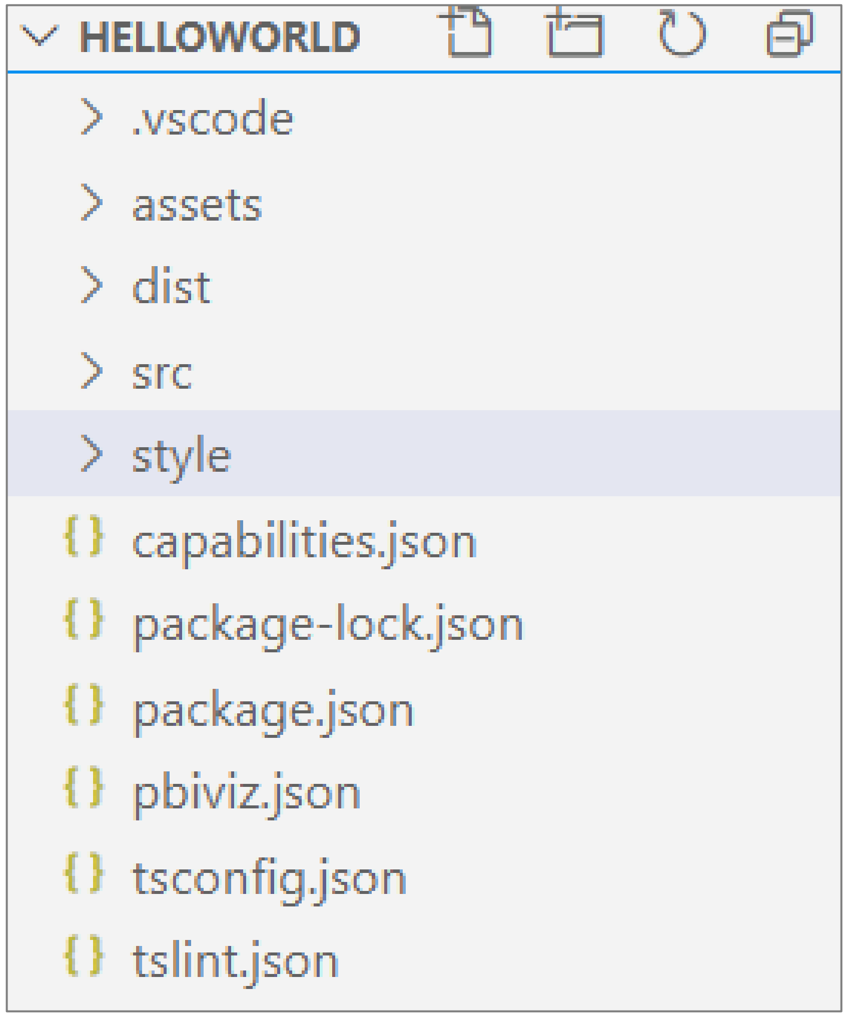
```
c:\Demo>pbiviz new HelloWorld
 info   Creating new visual
 done   Visual creation complete

c:\Demo>cd helloWorld

c:\Demo\helloWorld>code .

c:\Demo\helloWorld>_
```



```
.vscode
assets
dist
src
  settings.ts
  visual.ts
style
  visual.less
capabilities.json
package-lock.json
package.json
pbiviz.json
tsconfig.json
tslint.json
```

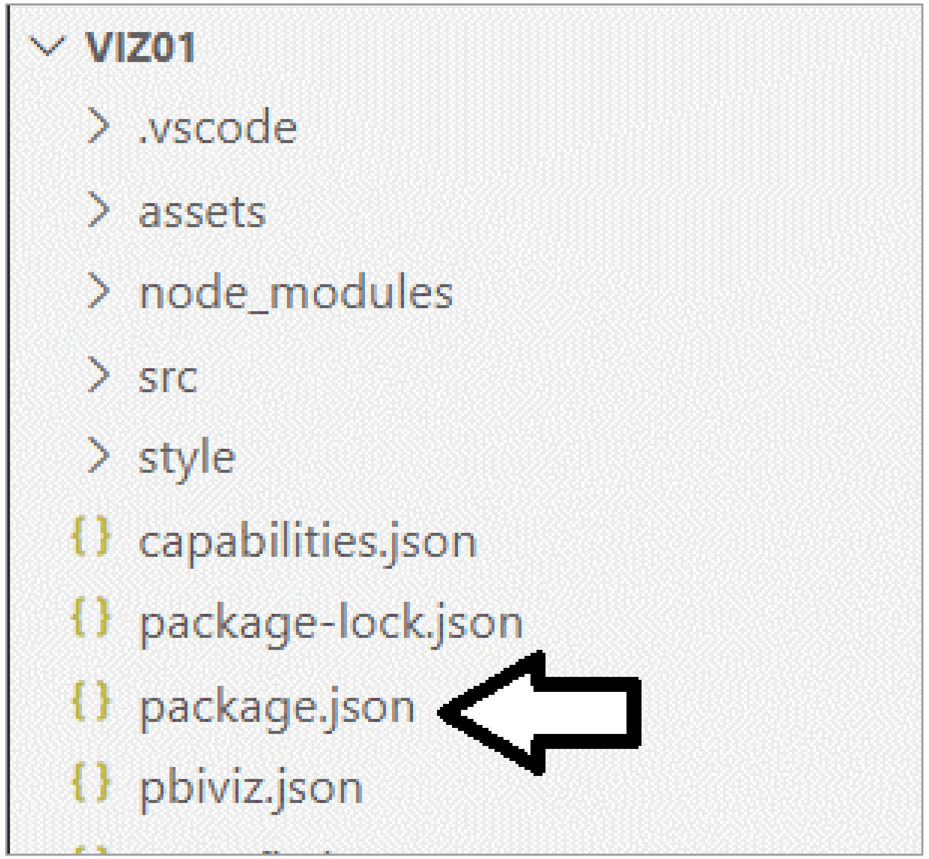
Top-level project files

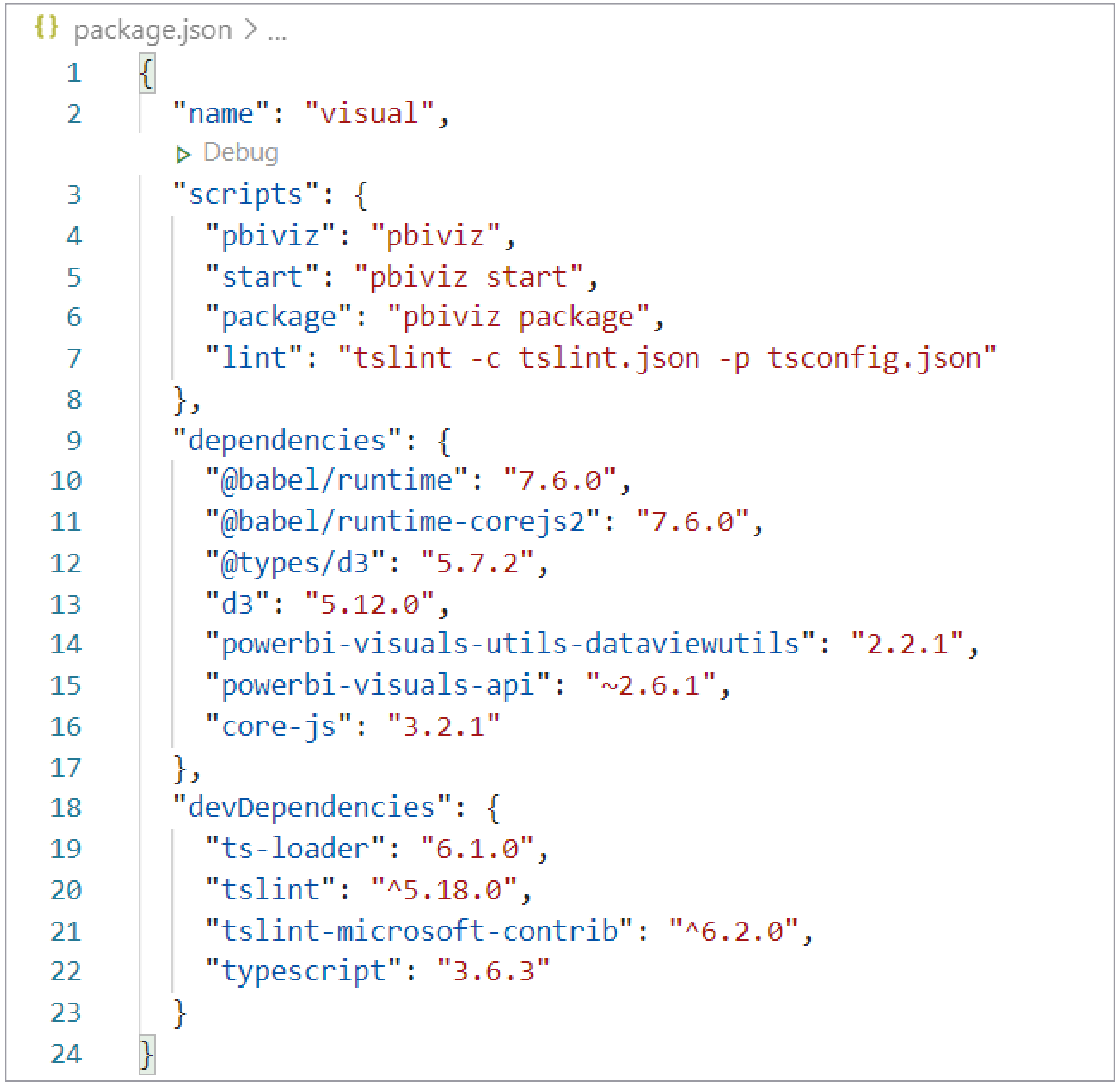
- **package.json**
  - Used by npm to manage packages
- **pbiviz.json**
  - Main manifest file for your custom visual project
- **capabilities.json**
  - File used to define visual capabilities
- **tsconfig.json & tslint.json**
  - Typescript compiler settings



```
HELLOWORLD
> .vscode
> assets
> dist
> src
> style
{} capabilities.json
{} package-lock.json
{} package.json
{} pbiviz.json
{} tsconfig.json
{} tslint.json
```

Package.json

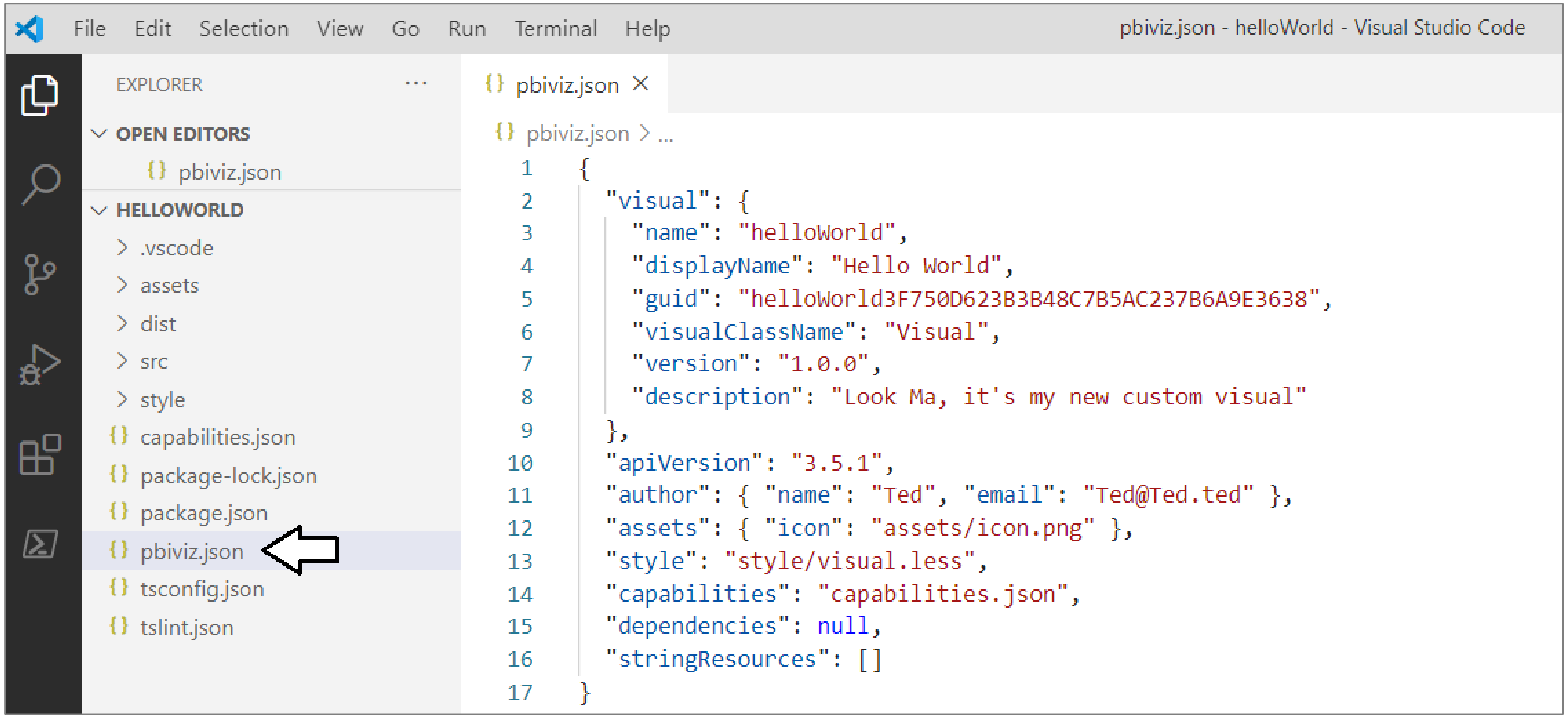




```json
{
  "name": "visual",
  "scripts": {
    "pbiviz": "pbiviz",
    "start": "pbiviz start",
    "package": "pbiviz package",
    "lint": "tslint -c tslint.json -p tsconfig.json"
  },
  "dependencies": {
    "@babel/runtime": "7.6.0",
    "@babel/runtime-corejs2": "7.6.0",
    "@types/d3": "5.7.2",
    "d3": "5.12.0",
    "powerbi-visuals-utils-dataviewutils": "2.2.1",
    "powerbi-visuals-api": "~2.6.1",
    "core-js": "3.2.1"
  },
  "devDependencies": {
    "ts-loader": "6.1.0",
    "tslint": "^5.18.0",
    "tslint-microsoft-contrib": "^6.2.0",
    "typescript": "3.6.3"
  }
}
```

The pbiviz.json File

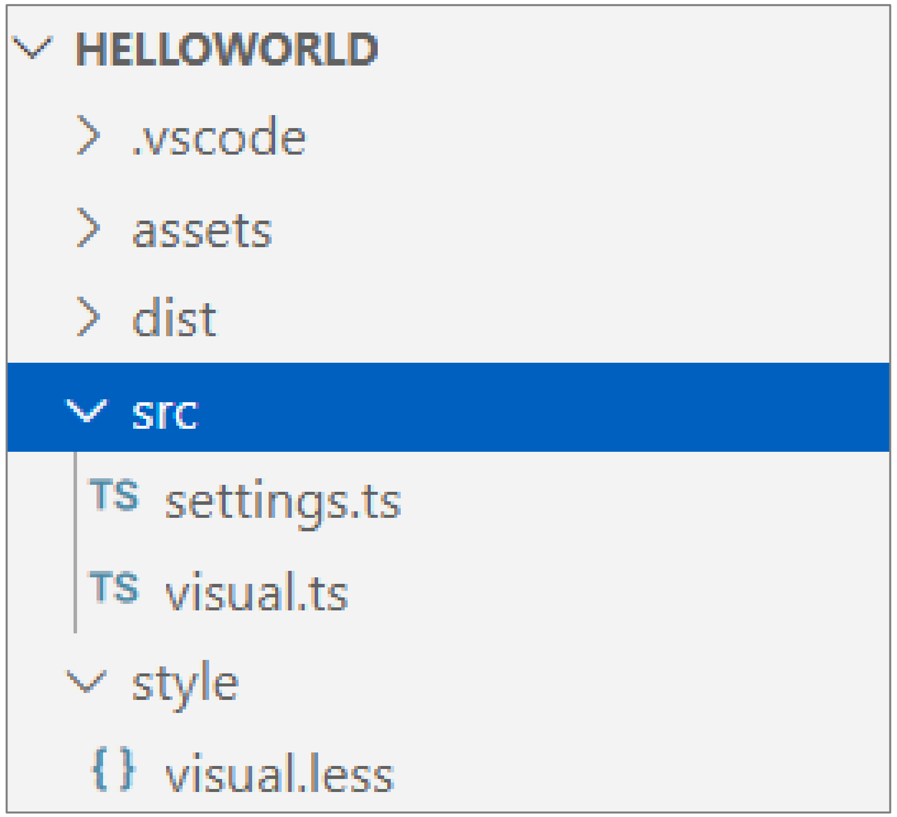
- **Acts as top-level manifest file for custom visual project**
  - Contains information used in final visual PBIVIZ packaging process



```json
{
  "visual": {
    "name": "helloWorld",
    "displayName": "Hello World",
    "guid": "helloWorld3F750D623B3B48C7B5AC237B6A9E3638",
    "visualClassName": "Visual",
    "version": "1.0.0",
    "description": "Look Ma, it's my new custom visual"
  },
  "apiVersion": "3.5.1",
  "author": { "name": "Ted", "email": "Ted@Ted.ted" },
  "assets": { "icon": "assets/icon.png" },
  "style": "style/visual.less",
  "capabilities": "capabilities.json",
  "dependencies": null,
  "stringResources": []
}
```

Visual Source Files

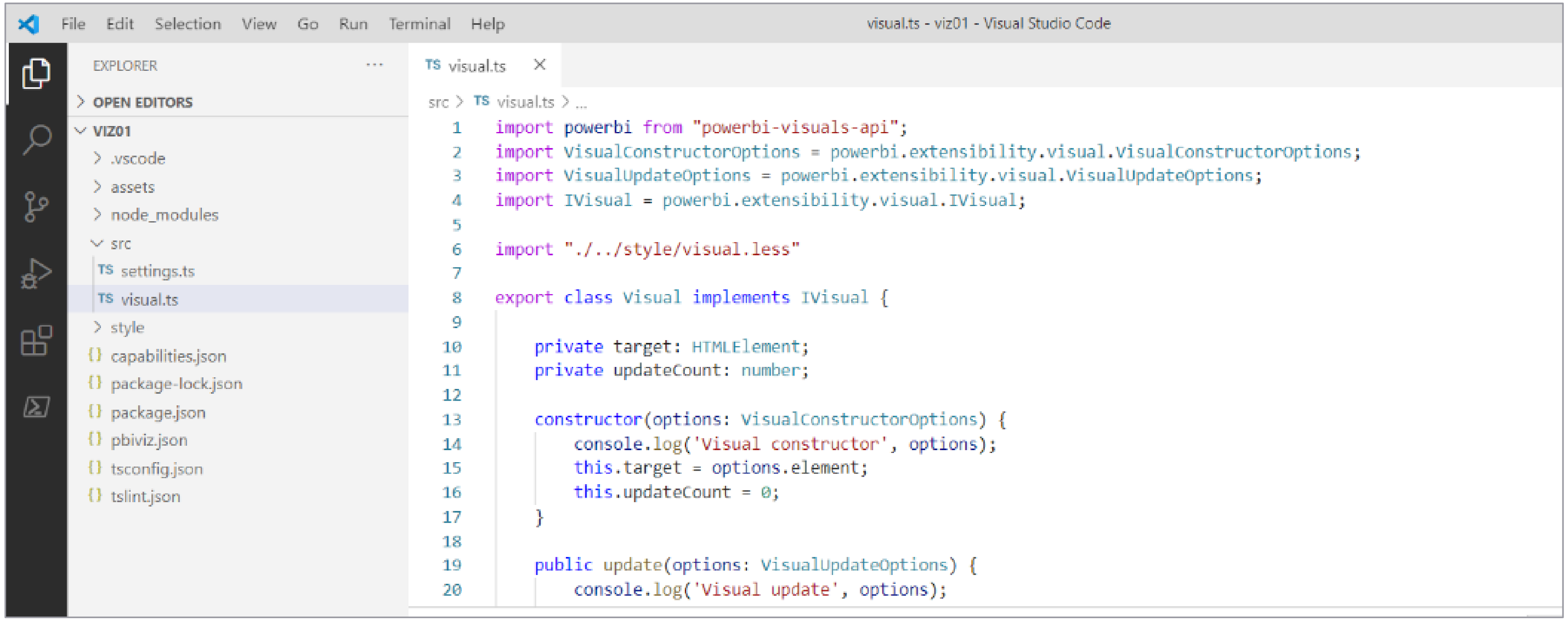
- src/visual.ts
  - visual class definition
- src/settings.ts
  - helper class to manage visual properties
- style/visual.less
  - CSS used to style custom visual



```
HELLOWORLD
  > .vscode
  > assets
  > dist
  v src
      TS settings.ts
      TS visual.ts
  v style
      {} visual.less
```

Authoring a Custom Visual Class

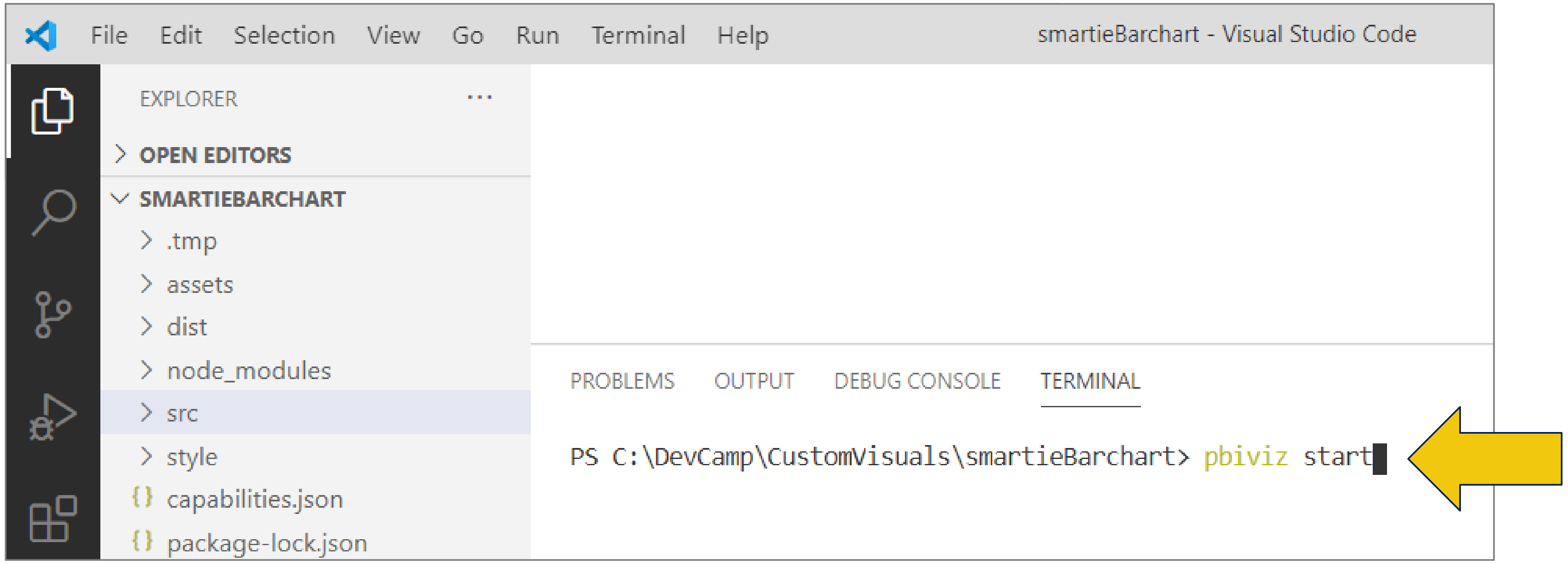
- **Custom visual is a class that implements IVisual**
  - Minimum visual class must implement **`IVisual`** interface and provide **`update`** method
  - Parameterized constructor used to create visual elements
  - **`update`** method performs visual rendering



```
import powerbi from "powerbi-visuals-api";
import VisualConstructorOptions = powerbi.extensibility.visual.VisualConstructorOptions;
import VisualUpdateOptions = powerbi.extensibility.visual.VisualUpdateOptions;
import IVisual = powerbi.extensibility.visual.IVisual;

import "./../style/visual.less"

export class Visual implements IVisual {

    private target: HTMLElement;
    private updateCount: number;

    constructor(options: VisualConstructorOptions) {
        console.log('Visual constructor', options);
        this.target = options.element;
        this.updateCount = 0;
    }

    public update(options: VisualUpdateOptions) {
        console.log('Visual update', options);
```

Visual Studio Code Terminal

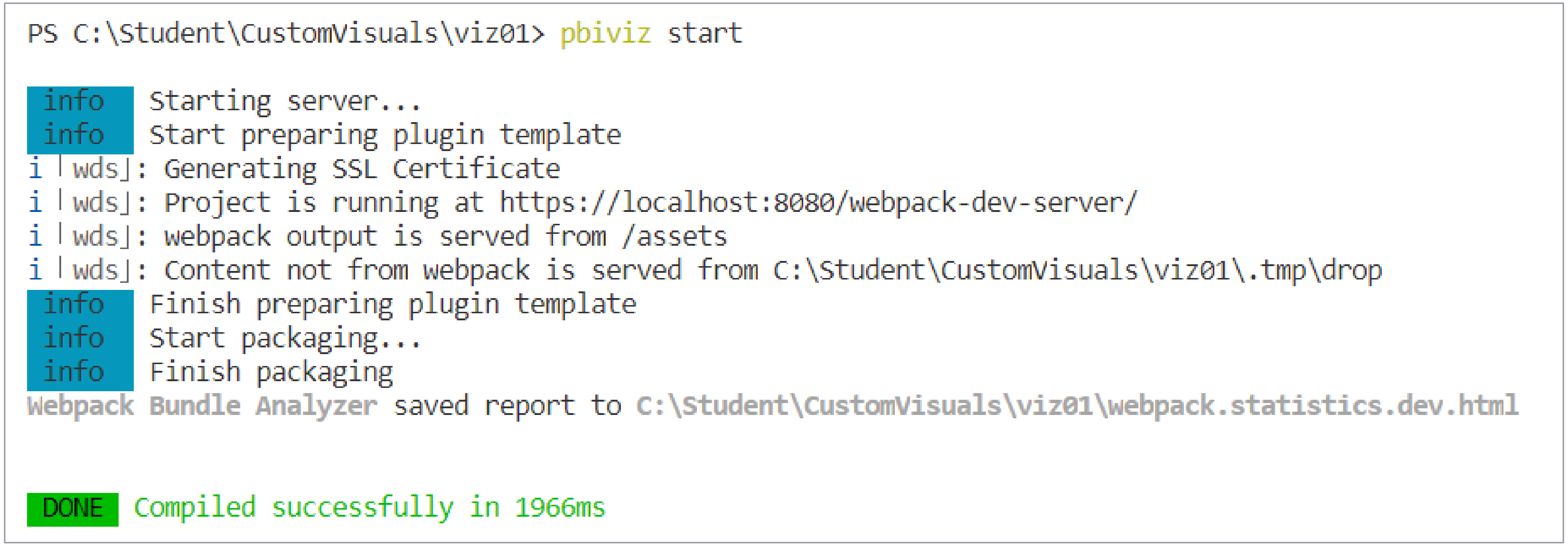
- Use the Terminal to execute commands with **npm** and **pbiviz**



```
PS C:\DevCamp\CustomVisuals\smartieBarchart> pbiviz start
```

Running a Custom Visual Project

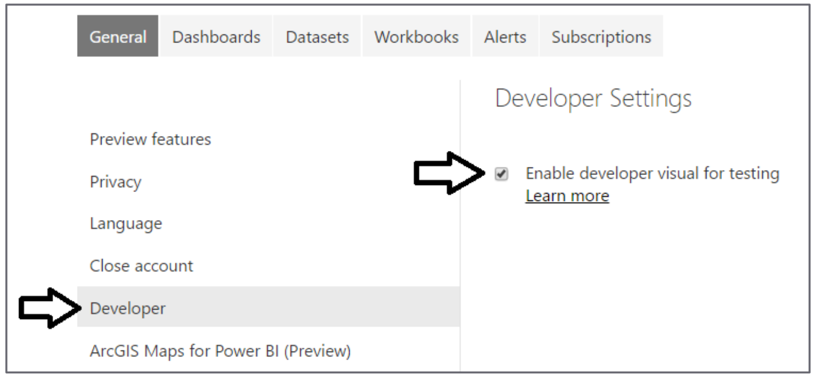
- **Visual projects run & tested using pbiviz start command**
  - Run pbiviz start from Visual Studio Code from Integrated console
  - Command starts local debugging session in node.js



```
PS C:\Student\CustomVisuals\viz01> pbiviz start

 info   Starting server...
 info   Start preparing plugin template
i ｢wds｣: Generating SSL Certificate
i ｢wds｣: Project is running at https://localhost:8080/webpack-dev-server/
i ｢wds｣: webpack output is served from /assets
i ｢wds｣: Content not from webpack is served from C:\Student\CustomVisuals\viz01\.tmp\drop
 info   Finish preparing plugin template
 info   Start packaging...
 info   Finish packaging
Webpack Bundle Analyzer saved report to C:\Student\CustomVisuals\viz01\webpack.statistics.dev.html


 DONE  Compiled successfully in 1966ms
```

The Developer Visual

- Must be enabled on Developer Settings page



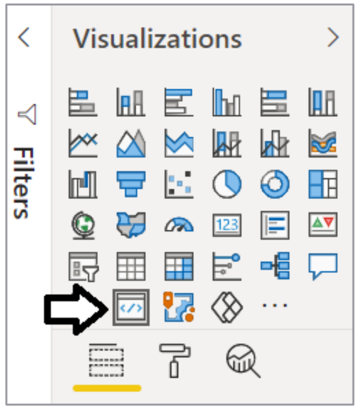
General Dashboards Datasets Workbooks Alerts Subscriptions

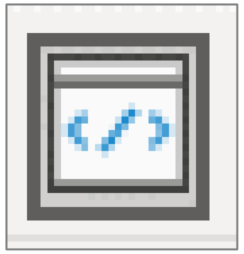
Preview features
Privacy
Language
Close account
Developer
ArcGIS Maps for Power BI (Preview)

Developer Settings

☑ Enable developer visual for testing
Learn more

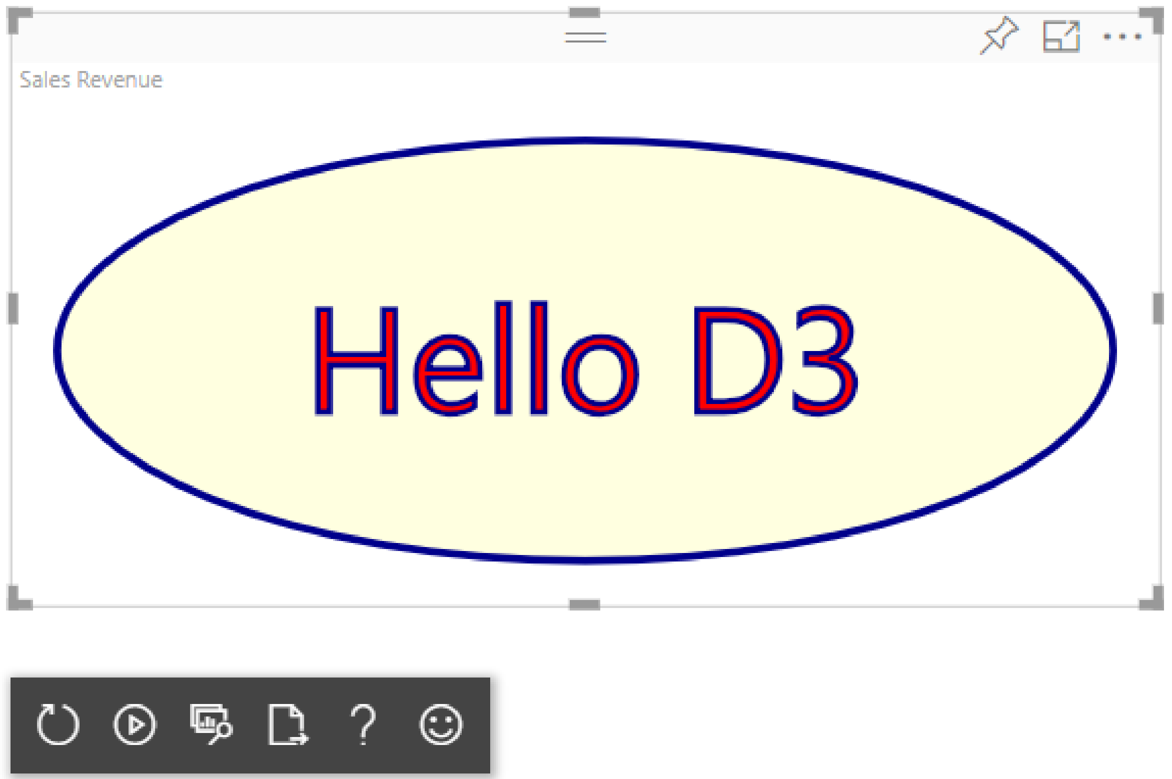
- Provides new Developer visual for testing and debugging custom visuals





Working with the Developer Visual

- **Developer visual loads custom visual from node.js**
  - Makes it possible to test custom visual inside Power BI Service
  - Developer visual provides toolbar with development utilities

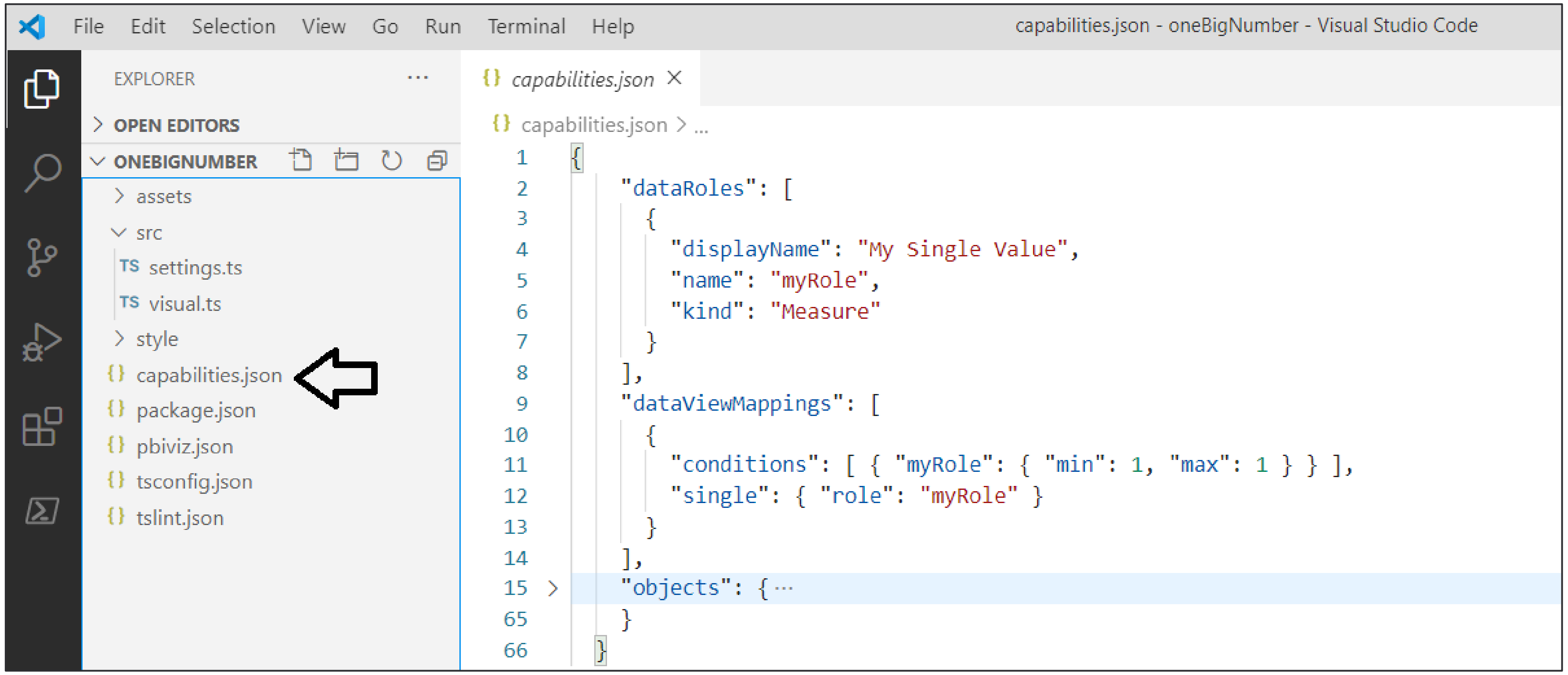


Agenda

- ✓ Installing the Power BI Developer Tools
- ✓ Creating Your First Custom Visual
- ➢ Defining Data Roles and Data Mappings
- • Extending a Visual with Custom Properties
- • Implementing Highlighting with SelectionManager
- • Custom Visual Packaging and Distribution

Visual Capabilities

- **Visual capabilities defined inside capabilities.json**
  - `dataRoles` define the field wells displayed on Fields pane
  - `dataViewMappings` define the type of DataView used by visual
  - `objects` created to define custom properties support by visual

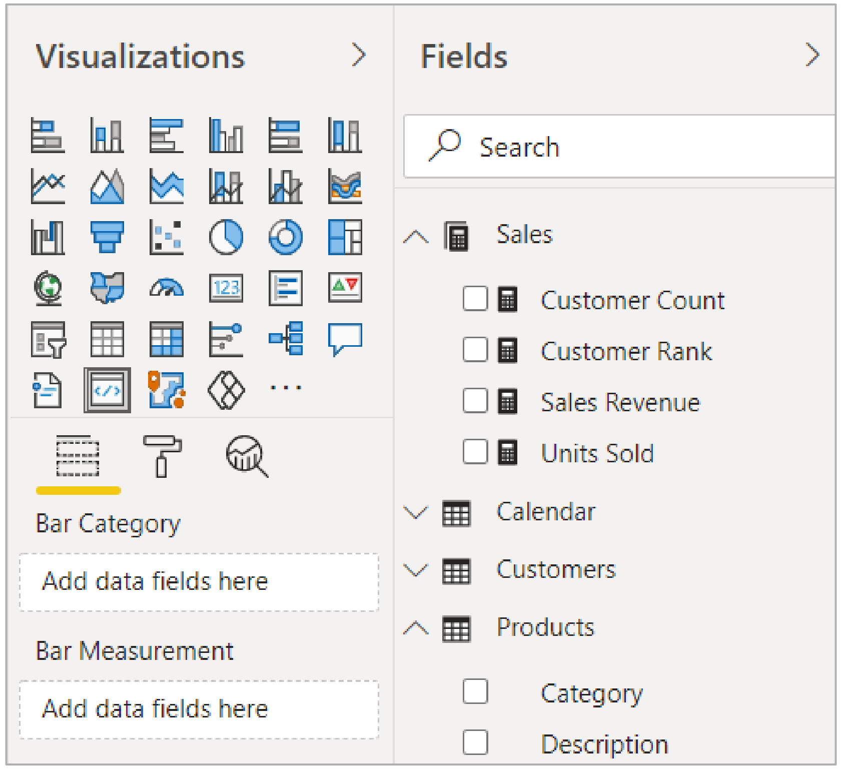


```
{
  "dataRoles": [
    {
      "displayName": "My Single Value",
      "name": "myRole",
      "kind": "Measure"
    }
  ],
  "dataViewMappings": [
    {
      "conditions": [ { "myRole": { "min": 1, "max": 1 } } ],
      "single": { "role": "myRole" }
    }
  ],
  "objects": {···
  }
}
```

Data Roles

- **DataRoles define how fields are associated with visual**
  - Each dataRole is display as field well in the Field pane
  - dataRoles can be defined with conditions and data mappings

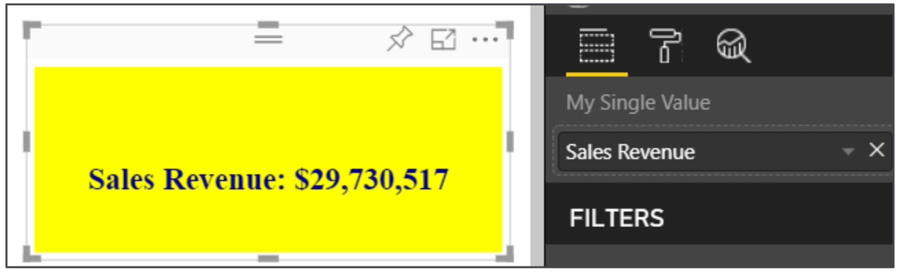
```
"dataRoles": [
  {
    "displayName": "Bar Grouping",
    "name": "myCategory",
    "kind": "Grouping"
  },
  {
    "displayName": "Bar Measurement",
    "name": "myMeasure",
    "kind": "Measure"
  }
]
```



Data Mapping Modes

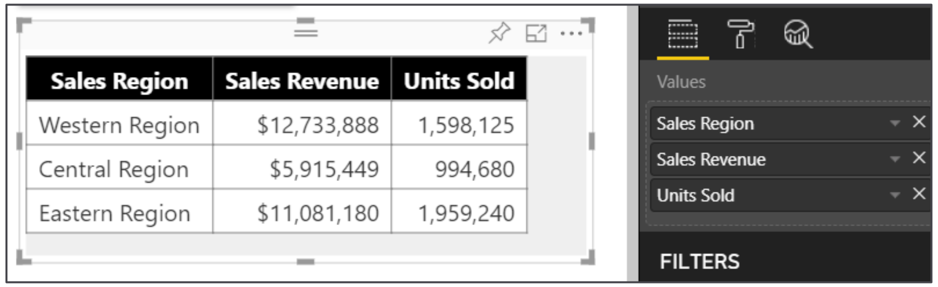
- Power BI visual API provides several mapping modes
  - Single
  - Table
  - Categorical
  - Matrix
  - Tree

Single Mapping



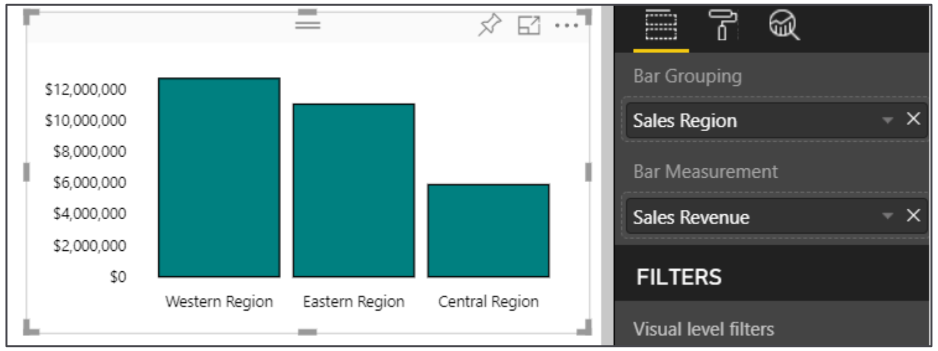
Sales Revenue: $29,730,517

Table Mapping



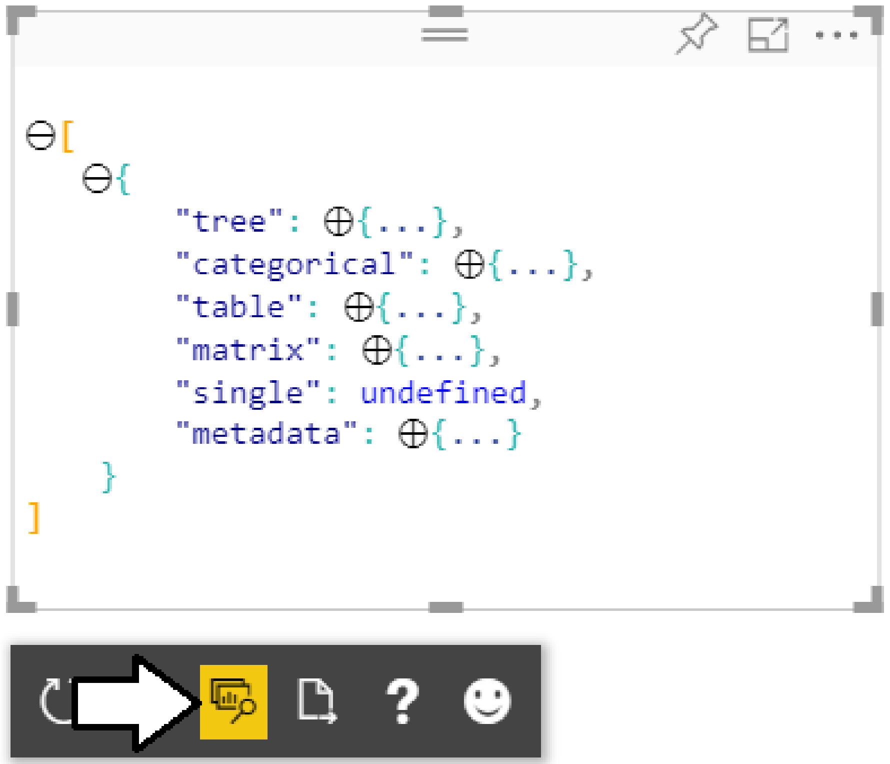
Sales Region	Sales Revenue	Units Sold
Western Region	$12,733,888	1,598,125
Central Region	$5,915,449	994,680
Eastern Region	$11,081,180	1,959,240

Categorical Mapping



Developer Visual DataView

- **Developer visual supports DataView mode**
  - Allows you to see and explore data mapping
  - Allows you to see metadata for custom properties

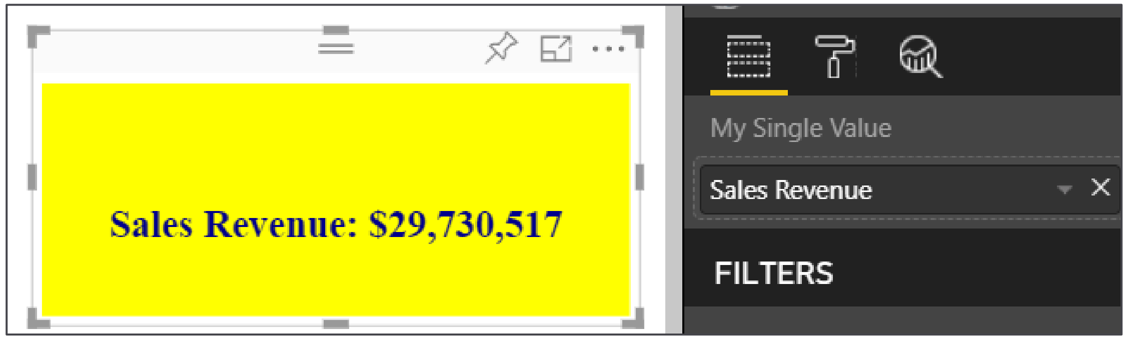


```
⊖[
   ⊖{
        "tree": ⊕{...},
        "categorical": ⊕{...},
        "table": ⊕{...},
        "matrix": ⊕{...},
        "single": undefined,
        "metadata": ⊕{...}
    }
]
```

Single Mapping Example: oneBigNumber

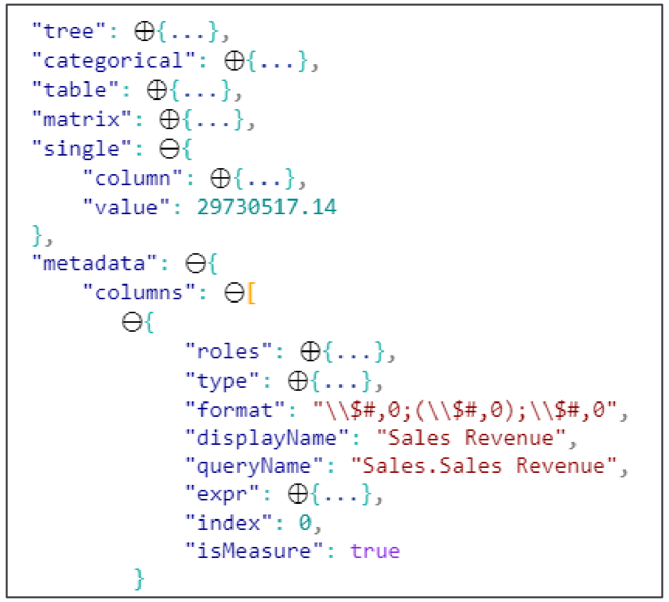
- **dataRole can use dataViewMapping mode of single**
  - For visuals like Card which only display single value
  - Condition can define that a dataRole requires exactly one measure

```
"dataRoles": [
  {
    "displayName": "My Single Value",
    "name": "myvalue",
    "kind": "Measure"
  }
],
"dataViewMappings": [
  {
    "conditions": [ { "myvalue": { "min": 1, "max": 1 } } ],
    "single": { "role": "myvalue" }
  }
]
```

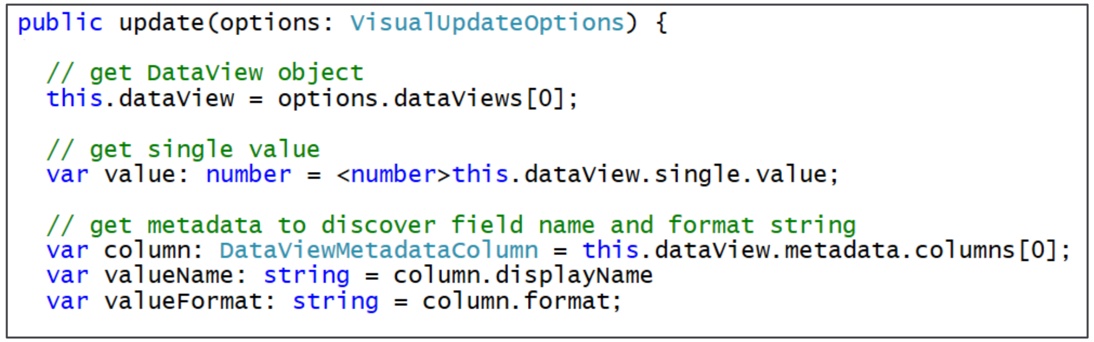

Sales Revenue: $29,730,517

Programming in Single Mapping Mode

- **Single mapping easy to access through visuals API**
  - DataView object provides single.value property
  - value property defined as PrimativeValue { bool | number | string }
  - PrimativeValue must be explicitly cast
  - Other measure properties available through column metadata



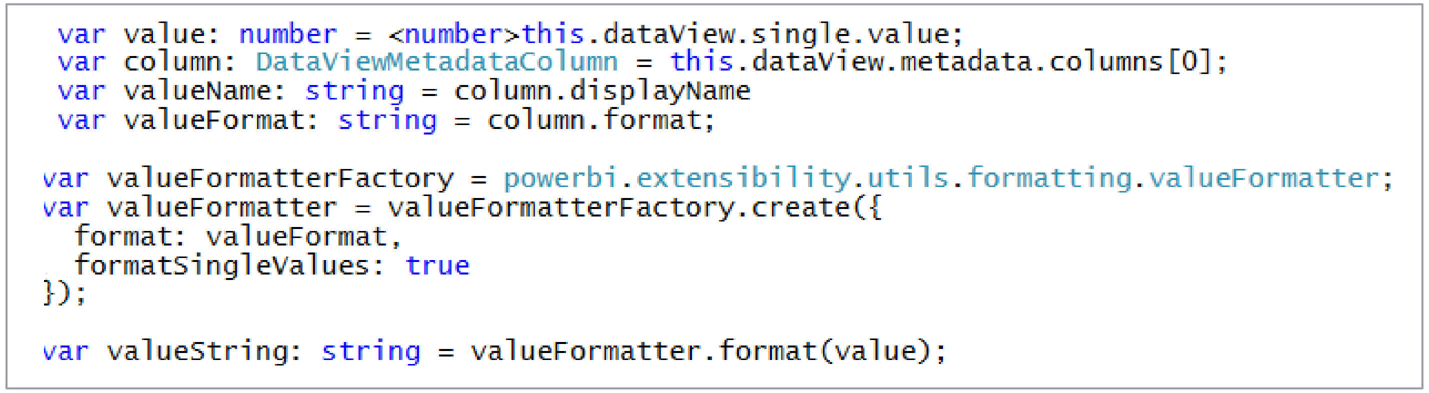
```
"tree": ⊕{...},
"categorical": ⊕{...},
"table": ⊕{...},
"matrix": ⊕{...},
"single": ⊖{
    "column": ⊕{...},
    "value": 29730517.14
},
"metadata": ⊖{
    "columns": ⊖[
       ⊖{
            "roles": ⊕{...},
            "type": ⊕{...},
            "format": "\\$#,0;(\\$#,0);\\$#,0",
            "displayName": "Sales Revenue",
            "queryName": "Sales.Sales Revenue",
            "expr": ⊕{...},
            "index": 0,
            "isMeasure": true
        }
```



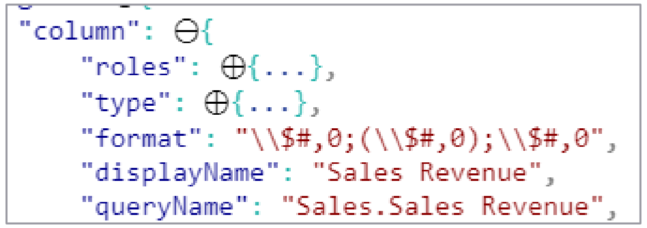
```
public update(options: VisualUpdateOptions) {

  // get DataView object
  this.dataView = options.dataViews[0];

  // get single value
  var value: number = <number>this.dataView.single.value;

  // get metadata to discover field name and format string
  var column: DataViewMetadataColumn = this.dataView.metadata.columns[0];
  var valueName: string = column.displayName
  var valueFormat: string = column.format;
```

Using the Power BI Formatting Utilities

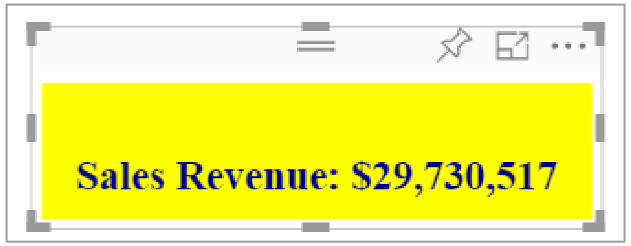
- **Used to format values using Power BI formatting strings**
  - Requires installing powerbi-visuals-utils-formattingutils package



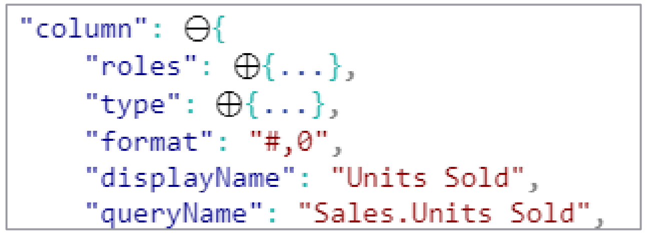
```
 var value: number = <number>this.dataView.single.value;
 var column: DataViewMetadataColumn = this.dataView.metadata.columns[0];
 var valueName: string = column.displayName
 var valueFormat: string = column.format;

var valueFormatterFactory = powerbi.extensibility.utils.formatting.valueFormatter;
var valueFormatter = valueFormatterFactory.create({
  format: valueFormat,
  formatSingleValues: true
});

var valueString: string = valueFormatter.format(value);
```



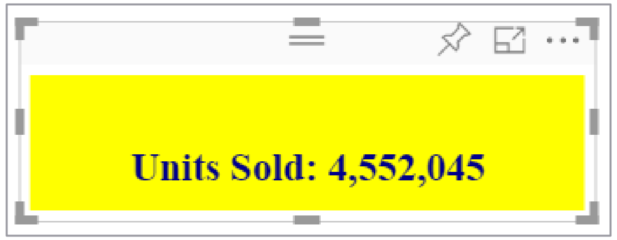
```
"column": ⊖{
    "roles": ⊕{...},
    "type": ⊕{...},
    "format": "\\$#,0;(\\$#,0);\\$#,0",
    "displayName": "Sales Revenue",
    "queryName": "Sales.Sales Revenue",
```



Sales Revenue: $29,730,517



```
"column": ⊖{
    "roles": ⊕{...},
    "type": ⊕{...},
    "format": "#,0",
    "displayName": "Units Sold",
    "queryName": "Sales.Units Sold",
```

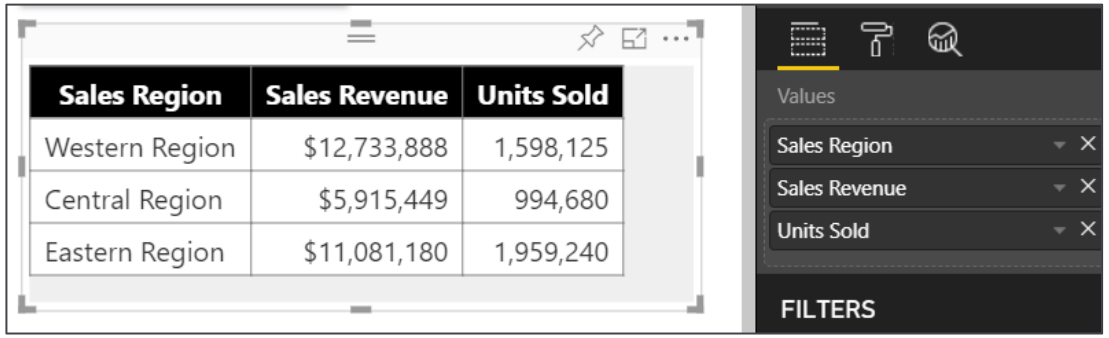


Units Sold: 4,552,045

Table Mapping Example: Snazzy Table

- **dataRole can use dataViewMapping mode of table**
  - For visuals which display rows & columns for ordered set of fields
  - condition can define number of fields that can be added

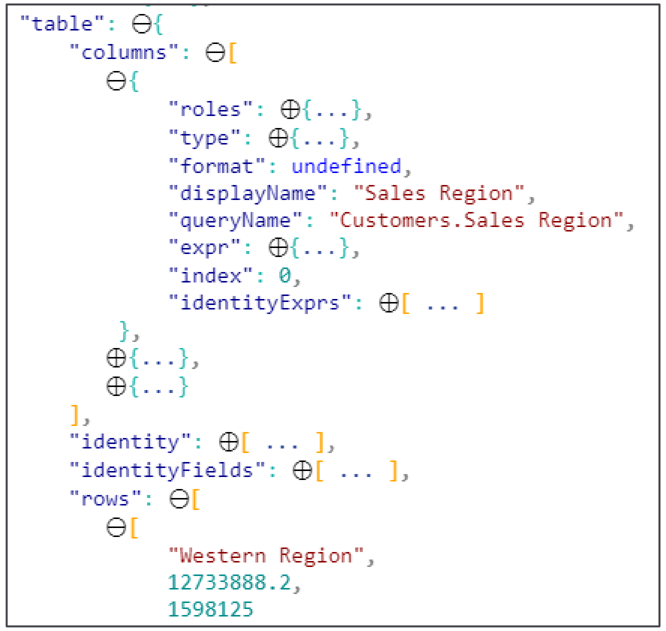
```
"dataRoles": [
   {
     "displayName": "Values",
     "name": "values",
     "kind": "GroupingOrMeasure"
   }
],
"dataViewMappings": [
   {
     "conditions": [ {"values": { "min": 1, "max": 5 } } ],
     "table": { "rows": { "for": { "in": "values" } } }
   }
]
```



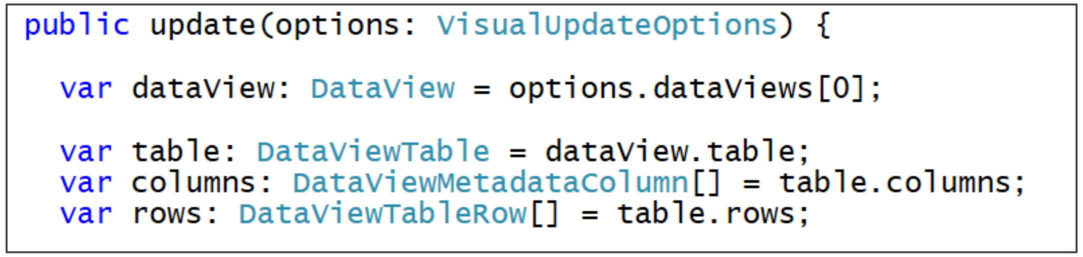
Sales Region	Sales Revenue	Units Sold
Western Region	$12,733,888	1,598,125
Central Region	$5,915,449	994,680
Eastern Region	$11,081,180	1,959,240

Programming in Table Mapping Mode

- **Table mapping data accessible through visuals API**
  - DataView object provides table property
  - table property provides columns property and rows property



```
"table": ⊖{
    "columns": ⊖[
       ⊖{
           "roles": ⊕{...},
           "type": ⊕{...},
           "format": undefined,
           "displayName": "Sales Region",
           "queryName": "Customers.Sales Region",
           "expr": ⊕{...},
           "index": 0,
           "identityExprs": ⊕[ ... ]
        },
       ⊕{...},
       ⊕{...}
    ],
    "identity": ⊕[ ... ],
    "identityFields": ⊕[ ... ],
    "rows": ⊖[
       ⊖[
           "Western Region",
           12733888.2,
           1598125
```

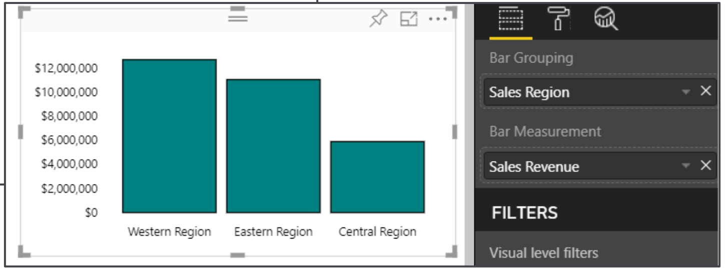


```
public update(options: VisualUpdateOptions) {

  var dataView: DataView = options.dataViews[0];

  var table: DataViewTable = dataView.table;
  var columns: DataViewMetadataColumn[] = table.columns;
  var rows: DataViewTableRow[] = table.rows;
```

Categorical Mapping Example: Barchart

- **dataRole can use dataViewMapping mode of categorical**
  - This is the most common type of data mapping
  - For visuals which divide data into groups for analysis
  - Groups defined as columns and values defined as measures

```
"dataRoles": [
  { "displayName": "Bar Grouping", "name": "myCategory", "kind": "Grouping" },
  { "displayName": "Bar Measurement", "name": "myMeasure", "kind": "Measure" }
],
"dataViewMappings": [
  {
    "conditions": [ { "myCategory": { "max": 1 }, "myMeasure": { "max": 1 } } ],
    "categorical": {
      "categories": {
        "for": { "in": "myCategory" },
        "dataReductionAlgorithm": { "top": {} }
      },
      "values": {
        "select": [ { "bind": { "to": "myMeasure" } } ]
      }
    }
  }
]
```

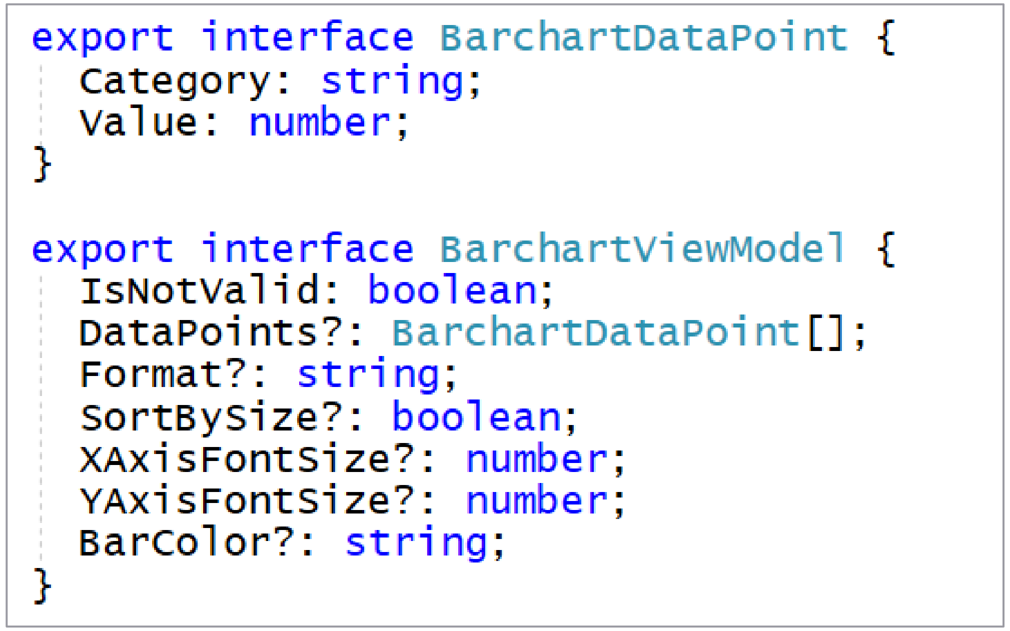


Bar Grouping: Sales Region

Bar Measurement: Sales Revenue

Designing with View Model

- **Best practice involves creating view model for each visual**
  - View model is a collection of data required for rendering visual
  - **`createViewModel`** method acquires data and constructs view model
  - **`update`** method calls **`createViewModel`** to get view model



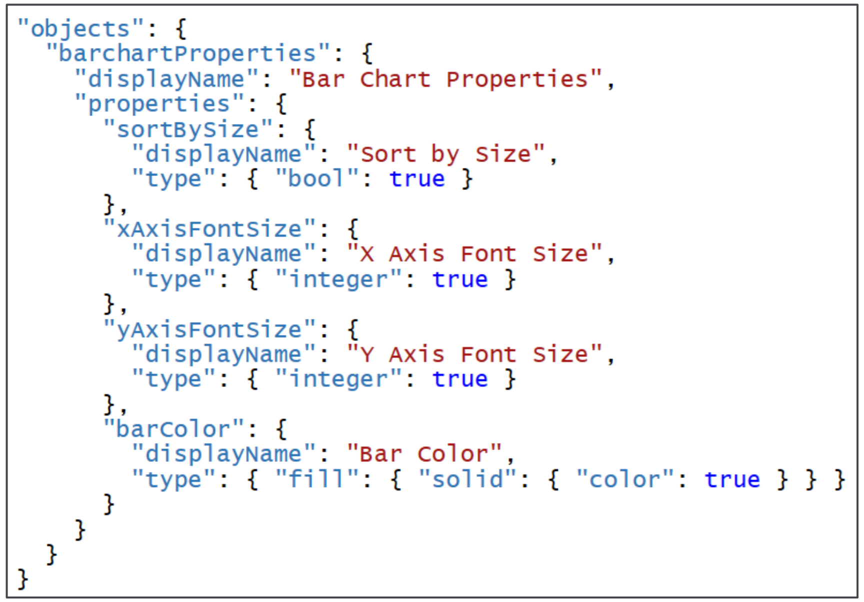
```
export interface BarchartDataPoint {
  Category: string;
  Value: number;
}

export interface BarchartViewModel {
  IsNotValid: boolean;
  DataPoints?: BarchartDataPoint[];
  Format?: string;
  SortBySize?: boolean;
  XAxisFontSize?: number;
  YAxisFontSize?: number;
  BarColor?: string;
}
```

Agenda

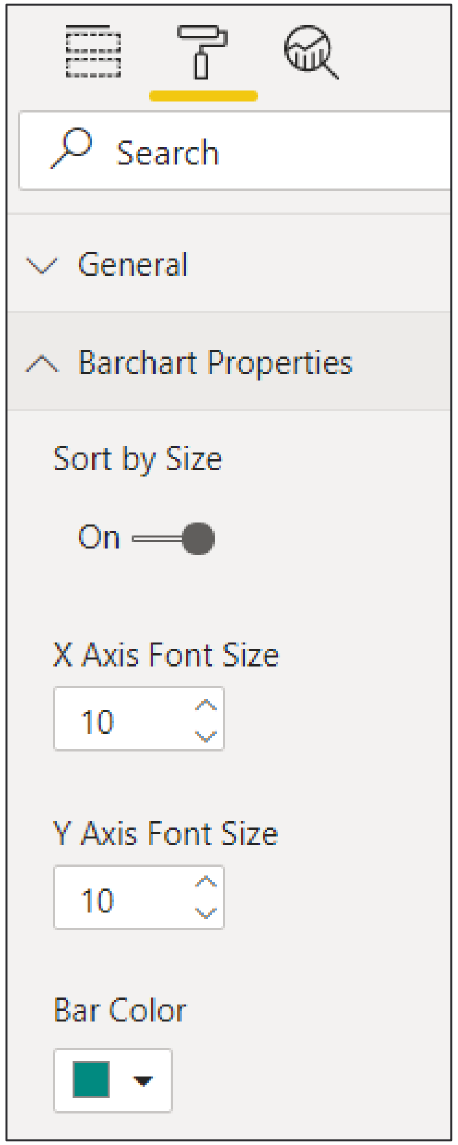
- ✓ Installing the Power BI Developer Tools
- ✓ Creating Your First Custom Visual
- ✓ Defining Data Roles and Data Mappings
- ➢ Extending a Visual with Custom Properties
- • Implementing Highlighting with SelectionManager
- • Custom Visual Packaging and Distribution

Extending Visuals with Custom Properties

- **Custom properties defined using objects**
  - You can define one or more objects in **capabilities.json**
  - Each object defined with name, display name and properties
  - object properties automatically persistent inside visual metadata
  - properties can be seen and modified by user in Format pane
  - Custom properties require extra code to initialize Format pane



```json
"objects": {
  "barchartProperties": {
    "displayName": "Bar Chart Properties",
    "properties": {
      "sortBySize": {
        "displayName": "Sort by Size",
        "type": { "bool": true }
      },
      "xAxisFontSize": {
        "displayName": "X Axis Font Size",
        "type": { "integer": true }
      },
      "yAxisFontSize": {
        "displayName": "Y Axis Font Size",
        "type": { "integer": true }
      },
      "barColor": {
        "displayName": "Bar Color",
        "type": { "fill": { "solid": { "color": true } } }
      }
    }
  }
}
```



Search

General

Barchart Properties

Sort by Size

On

X Axis Font Size

10

Y Axis Font Size

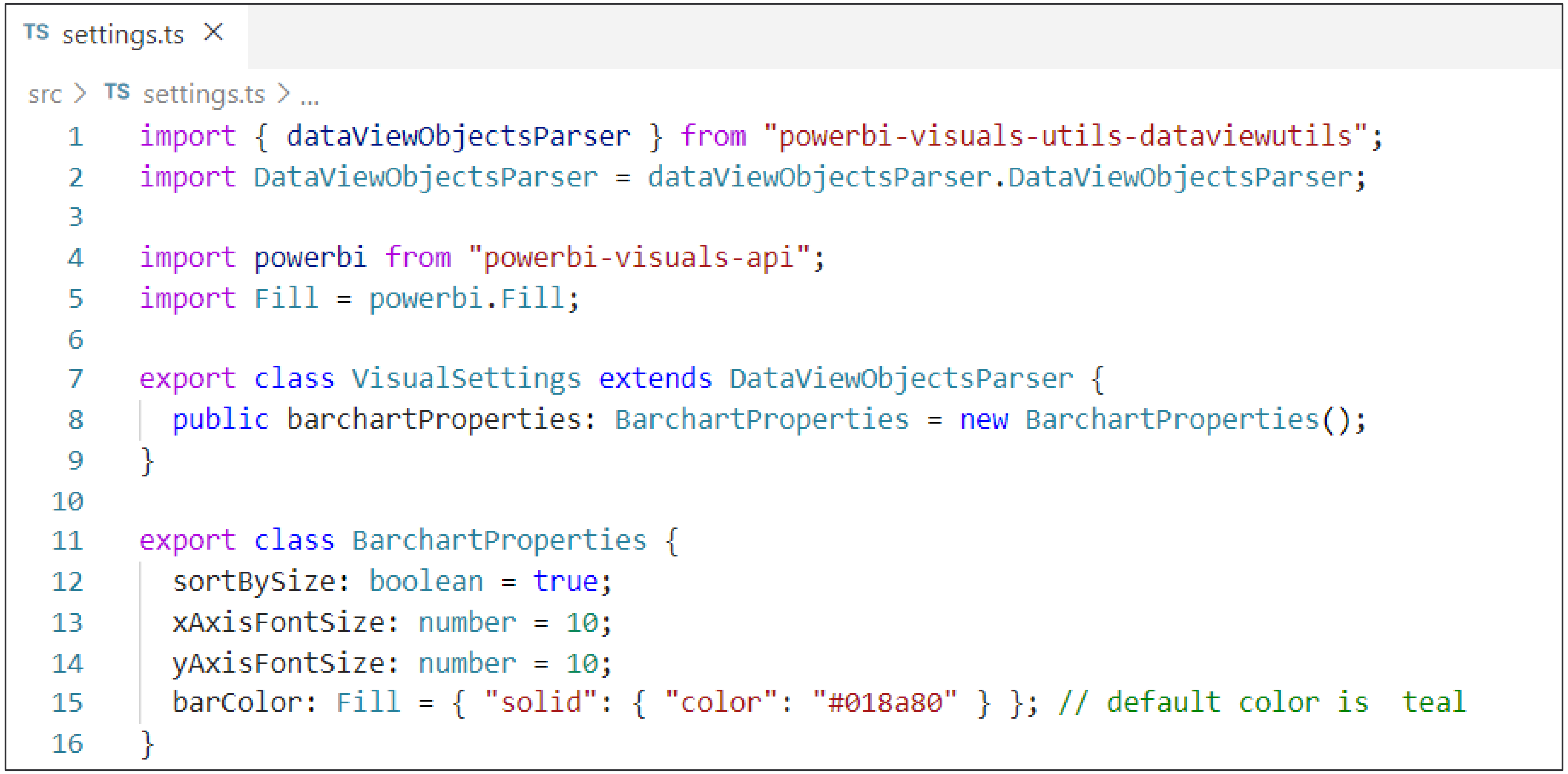
10

Bar Color

🟩

DataViewObjectParser and VisualSettings

- **Power BI visual utilities provide DataViewObjectParser**
  - Abstracts away tricky code to initialize and read property values



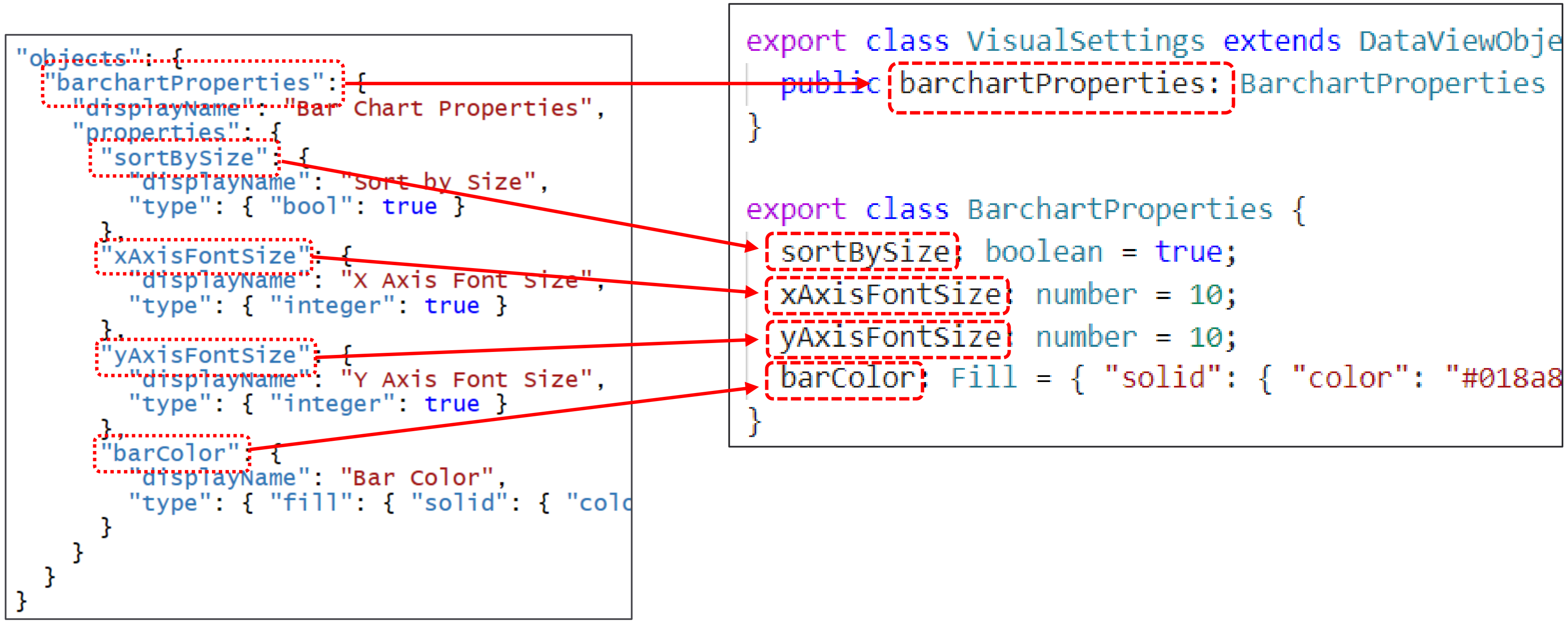
settings.ts

src > settings.ts > ...

```
import { dataViewObjectsParser } from "powerbi-visuals-utils-dataviewutils";
import DataViewObjectsParser = dataViewObjectsParser.DataViewObjectsParser;

import powerbi from "powerbi-visuals-api";
import Fill = powerbi.Fill;

export class VisualSettings extends DataViewObjectsParser {
  public barchartProperties: BarchartProperties = new BarchartProperties();
}

export class BarchartProperties {
  sortBySize: boolean = true;
  xAxisFontSize: number = 10;
  yAxisFontSize: number = 10;
  barColor: Fill = { "solid": { "color": "#018a80" } }; // default color is  teal
}
```

Mapping Object Properties to VisualSettings

- **VisualSettings class must map to named objectnamed**
  - VisualSetting class contains named field that maps to object name
  - Named field based on custom class with mapped properties
  - Object & property names must match what's in capabilities.json



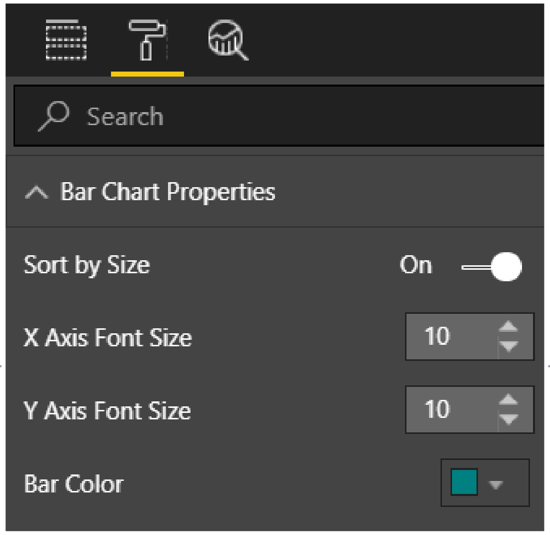
```
"objects": {
  "barchartProperties": {
    "displayName": "Bar Chart Properties",
    "properties": {
      "sortBySize": {
        "displayName": "Sort by Size",
        "type": { "bool": true }
      },
      "xAxisFontSize": {
        "displayName": "X Axis Font Size",
        "type": { "integer": true }
      },
      "yAxisFontSize": {
        "displayName": "Y Axis Font Size",
        "type": { "integer": true }
      },
      "barColor": {
        "displayName": "Bar Color",
        "type": { "fill": { "solid": { "colo
      }
    }
  }
}
```

```
export class VisualSettings extends DataViewObje
  public barchartProperties: BarchartProperties
}

export class BarchartProperties {
  sortBySize: boolean = true;
  xAxisFontSize: number = 10;
  yAxisFontSize: number = 10;
  barColor: Fill = { "solid": { "color": "#018a8
}
```

Initializing Objects in the Format Pane

- **Visual must initialize properties in Format pane**
  - Visual must implement enumerateObjectInstances
  - VisualSettings makes this relatively easy
  - Extra code required to make property appear as spinner

```
public enumerateObjectInstances(options: EnumerateVisualObjectInstancesOptions): VisualObjectInstanceEnumeration {

  // register object properties
  var visualObjects: VisualObjectInstanceEnumerationObject =
    <VisualObjectInstanceEnumerationObject>VisualSettings
      .enumerateObjectInstances(this.settings, options);

  // configure spinners for integers properties
  visualObjects.instances[0].validValues = {
    xAxisFontSize: { numberRange: { min: 10, max: 36 } },
    yAxisFontSize: { numberRange: { min: 10, max: 36 } },
  };

  // return visual object collection
  return visualObjects;
}
```



Bar Chart Properties

Sort by Size On

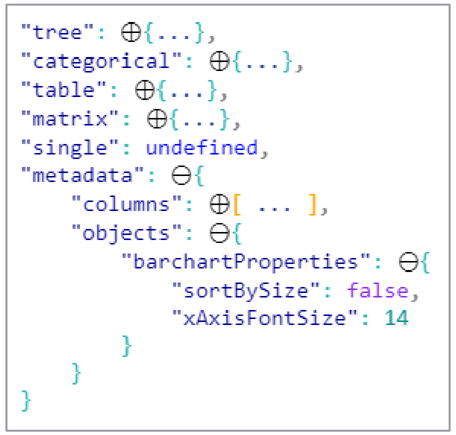
X Axis Font Size 10

Y Axis Font Size 10

Bar Color

Retrieving Property Values

- **Property values persisted into visual metadata**
  - Properties not persisted white they still retain default values



```
"tree": ⊕{...},
"categorical": ⊕{...},
"table": ⊕{...},
"matrix": ⊕{...},
"single": undefined,
"metadata": ⊖{
    "columns": ⊕[ ... ],
    "objects": ⊖{
        "barchartProperties": ⊖{
            "sortBySize": false,
            "xAxisFontSize": 14
        }
    }
}
```

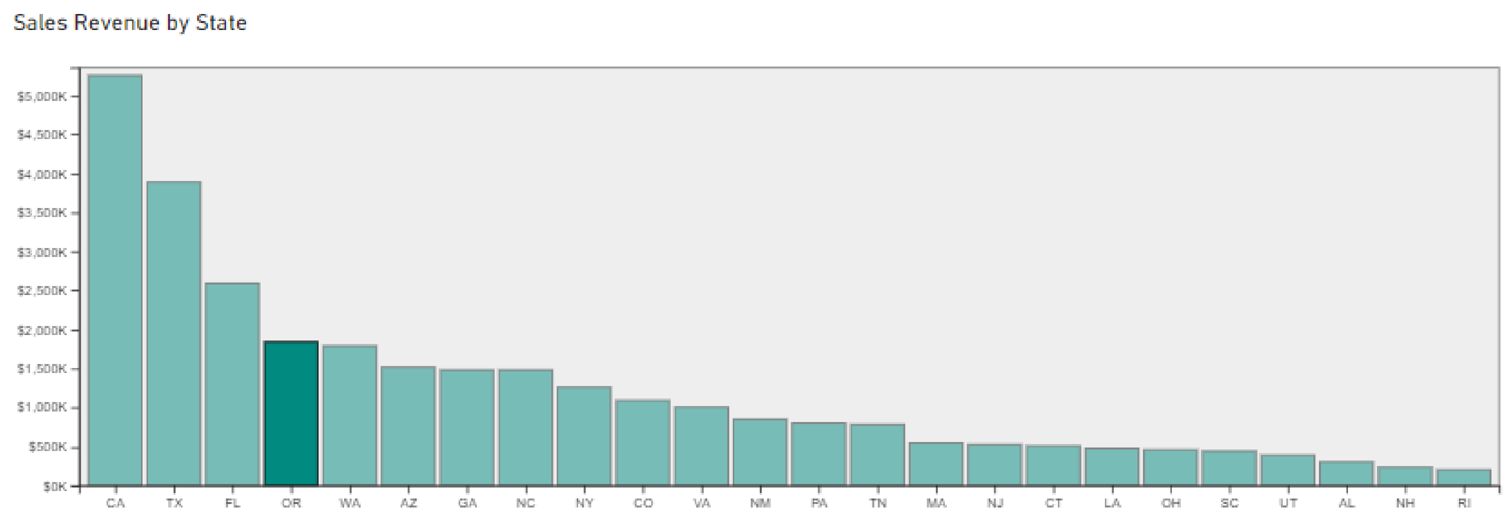
  - Property values retrieved using VisualSettings object

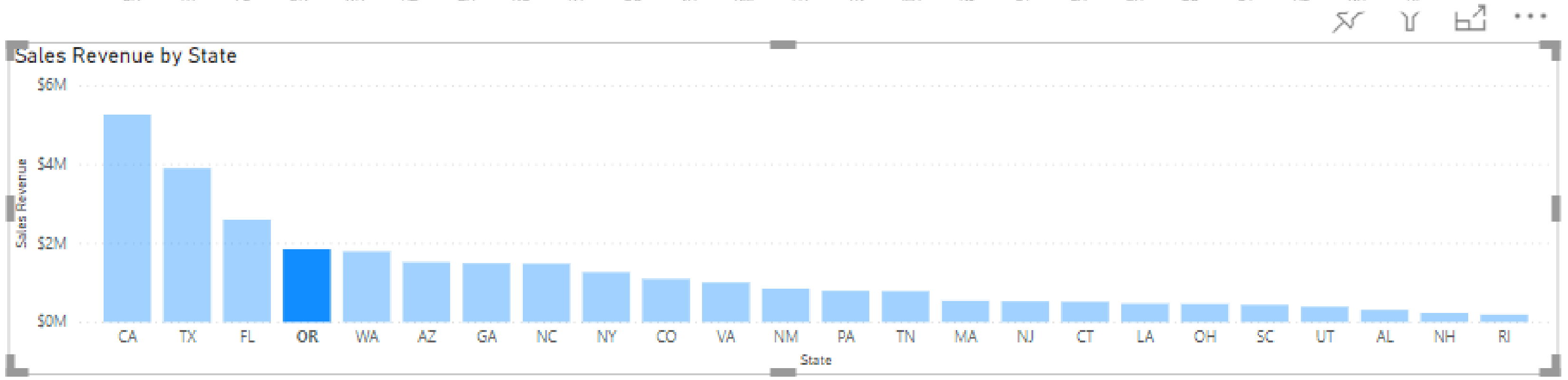
```
public update(options: VisualUpdateOptions) {

  if (options.dataViews[0]) {

    // create VisualSettings object
    this.settings = VisualSettings.parse(options.dataViews[0]) as VisualSettings;

    // retrieve property values
    var sortBySize: boolean = this.settings.barchartProperties.sortBySize
    var xAxisFontSize: number = this.settings.barchartProperties.xAxisFontSize;
```

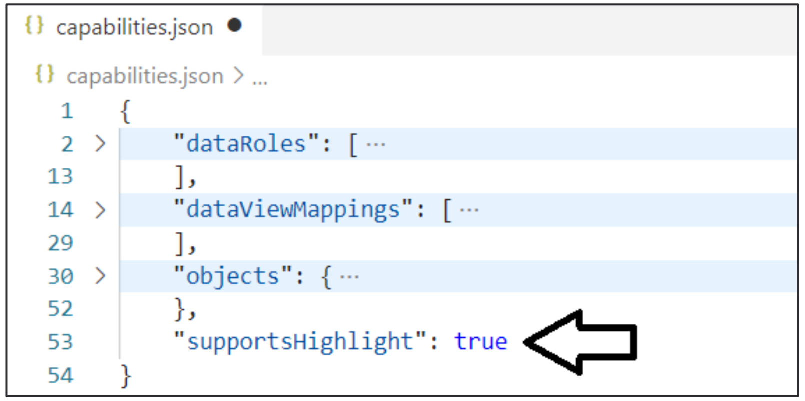
Agenda

- ✓ Installing the Power BI Developer Tools
- ✓ Creating Your First Custom Visual
- ✓ Defining Data Roles and Data Mappings
- ✓ Extending a Visual with Custom Properties
- ➢ Implementing Highlighting with SelectionManager
- • Custom Visual Packaging and Distribution

Implementing Visual Highlighting Support







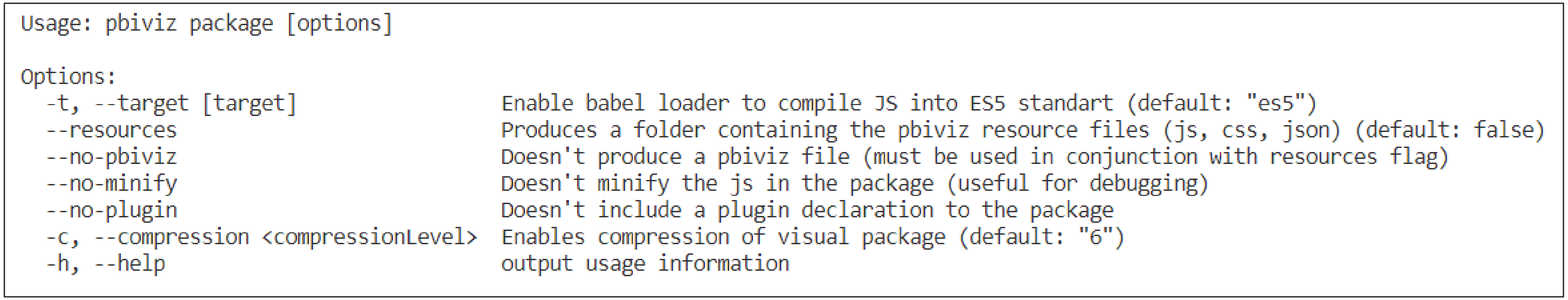
```
{
  "dataRoles": [ ···
  ],
  "dataViewMappings": [ ···
  ],
  "objects": { ···
  },
  "supportsHighlight": true
}
```

Agenda

- ✓ Installing the Power BI Developer Tools
- ✓ Creating Your First Custom Visual
- ✓ Defining Data Roles and Data Mappings
- ✓ Extending a Visual with Custom Properties
- ✓ Implementing Highlighting with SelectionManager
- ➢ Custom Visual Packaging and Distribution

Packaging A Custom Visual for Deployment

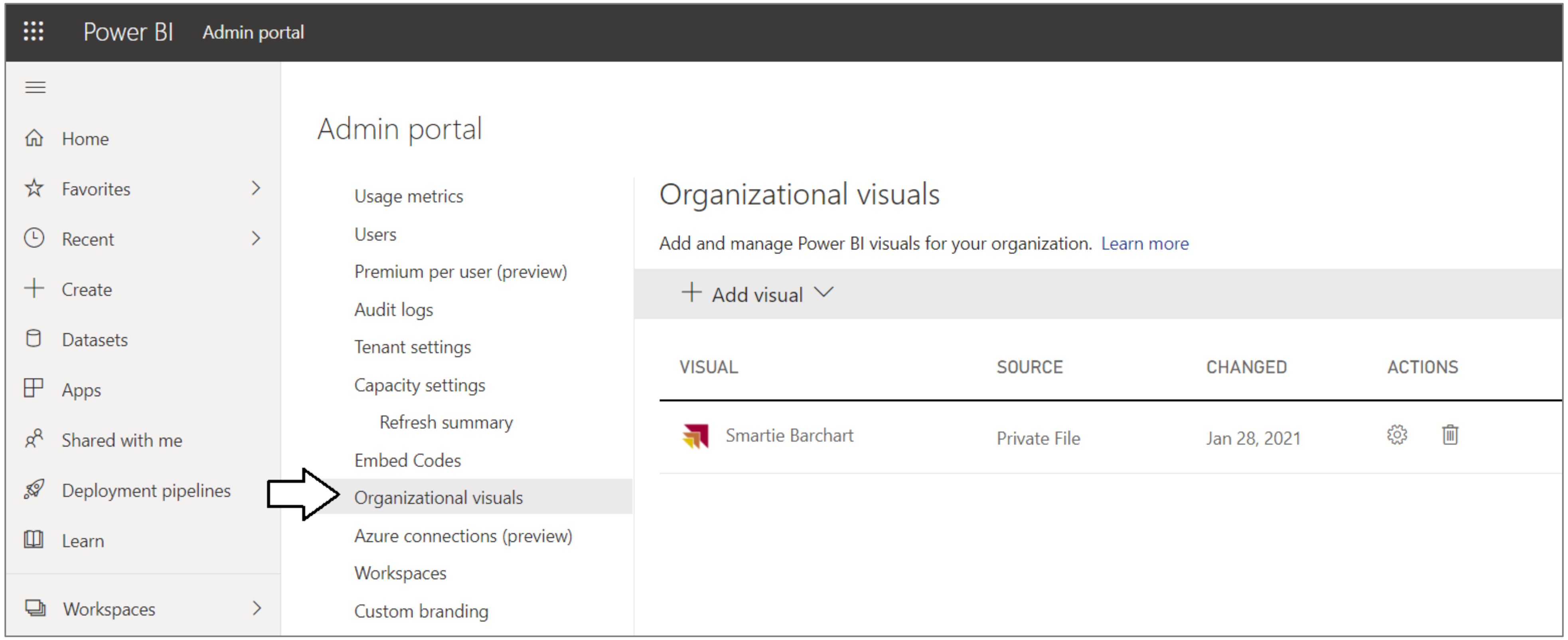
- **Use the `pbiviz package` command to build PBIVIZ file for distribution**
  - Build versioned package for distribution
  - Build version for testing in Power BI Desktop



```
Usage: pbiviz package [options]

Options:
  -t, --target [target]                 Enable babel loader to compile JS into ES5 standart (default: "es5")
  --resources                           Produces a folder containing the pbiviz resource files (js, css, json) (default: false)
  --no-pbiviz                           Doesn't produce a pbiviz file (must be used in conjunction with resources flag)
  --no-minify                           Doesn't minify the js in the package (useful for debugging)
  --no-plugin                           Doesn't include a plugin declaration to the package
  -c, --compression <compressionLevel>  Enables compression of visual package (default: "6")
  -h, --help                            output usage information
```

Organizational Visuals Gallery

- Make custom visuals available on organization-wide basis



Organizational visuals

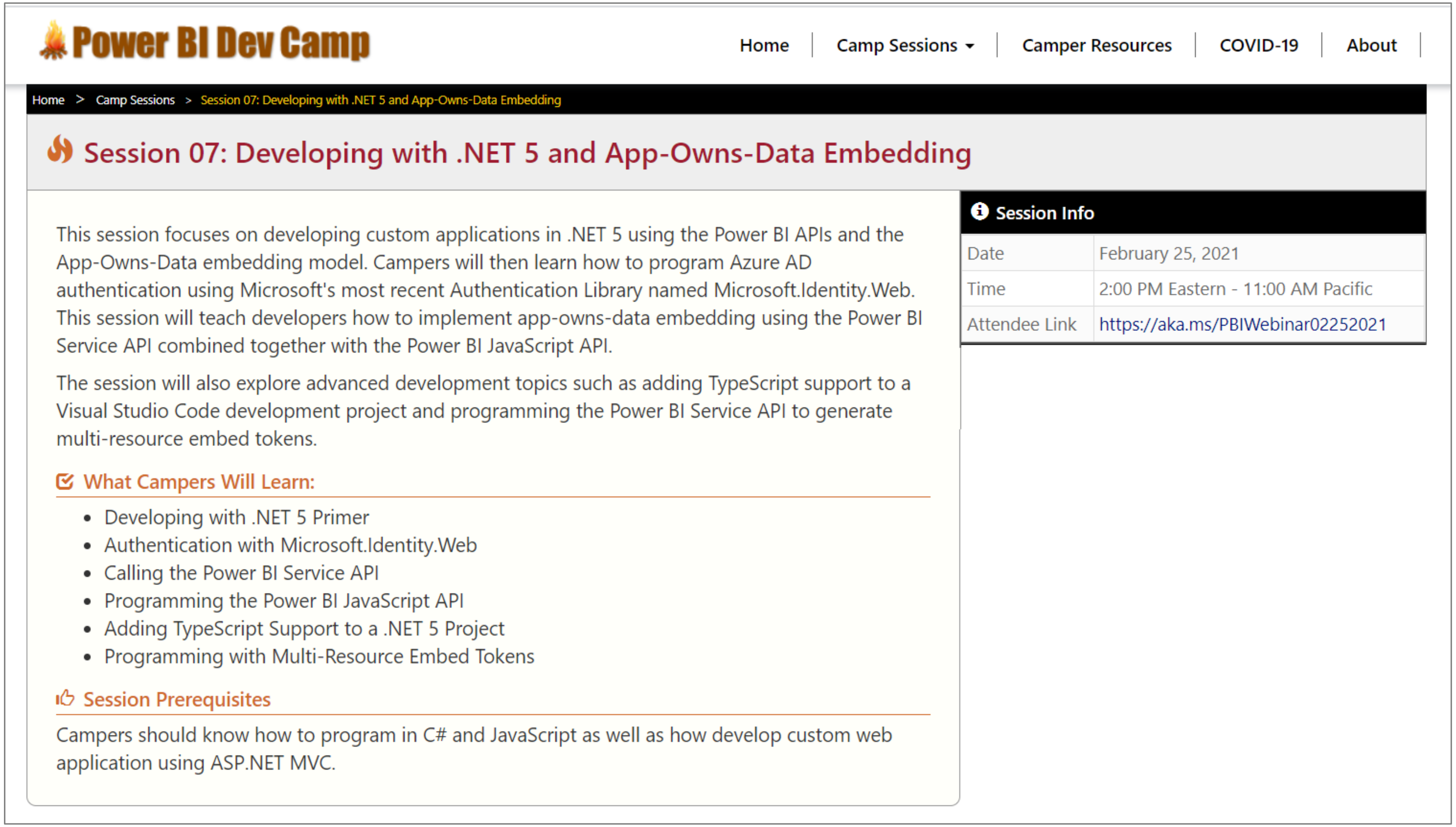
Add and manage Power BI visuals for your organization. Learn more

VISUAL	SOURCE	CHANGED	ACTIONS
Smartie Barchart	Private File	Jan 28, 2021	

Summary

- ✓ Installing the Power BI Developer Tools
- ✓ Creating Your First Custom Visual
- ✓ Defining Data Roles and Data Mappings
- ✓ Extending a Visual with Custom Properties
- ✓ Implementing Highlighting with SelectionManager
- ✓ Custom Visual Packaging and Distribution

Get Ready for Next Month...


Power BI Dev Camp
Home | Camp Sessions | Camper Resources | COVID-19 | About
Home > Camp Sessions > Session 07: Developing with .NET 5 and App-Owns-Data Embedding

Session 07: Developing with .NET 5 and App-Owns-Data Embedding

This session focuses on developing custom applications in .NET 5 using the Power BI APIs and the App-Owns-Data embedding model. Campers will then learn how to program Azure AD authentication using Microsoft's most recent Authentication Library named Microsoft.Identity.Web. This session will teach developers how to implement app-owns-data embedding using the Power BI Service API combined together with the Power BI JavaScript API.

The session will also explore advanced development topics such as adding TypeScript support to a Visual Studio Code development project and programming the Power BI Service API to generate multi-resource embed tokens.

What Campers Will Learn:

- Developing with .NET 5 Primer
- Authentication with Microsoft.Identity.Web
- Calling the Power BI Service API
- Programming the Power BI JavaScript API
- Adding TypeScript Support to a .NET 5 Project
- Programming with Multi-Resource Embed Tokens

Session Prerequisites

Campers should know how to program in C# and JavaScript as well as how develop custom web application using ASP.NET MVC.

Session Info

Date	February 25, 2021
Time	2:00 PM Eastern - 11:00 AM Pacific
Attendee Link	https://aka.ms/PBIWebinar02252021

Questions