

Getting Started with the Power BI Developer Tools



Agenda

- Custom Visuals in Power BI
- Node.JS and the Cross-platform Toolchain
- Creating Projects with the PBIVIZ CLI
- Custom Visual Project Structure
- Adding Typed Definition Files
- Testing and Debugging a Custom Visual



Install the Power BI Developer Toolchain

- Install Node.JS
 - Installs Node Package Manager (npm)
- Install Visual Studio Code
 - Lightweight Alternative to Visual Studio for Node.js Development
- Install Power BI visuals CLI tool (pbiviz)
 - Install using Node Package Manager (npm)
- Install Local self-signed certificate
 - Install using Power BI visuals CLI tool (pbiviz)



Installing node.js

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



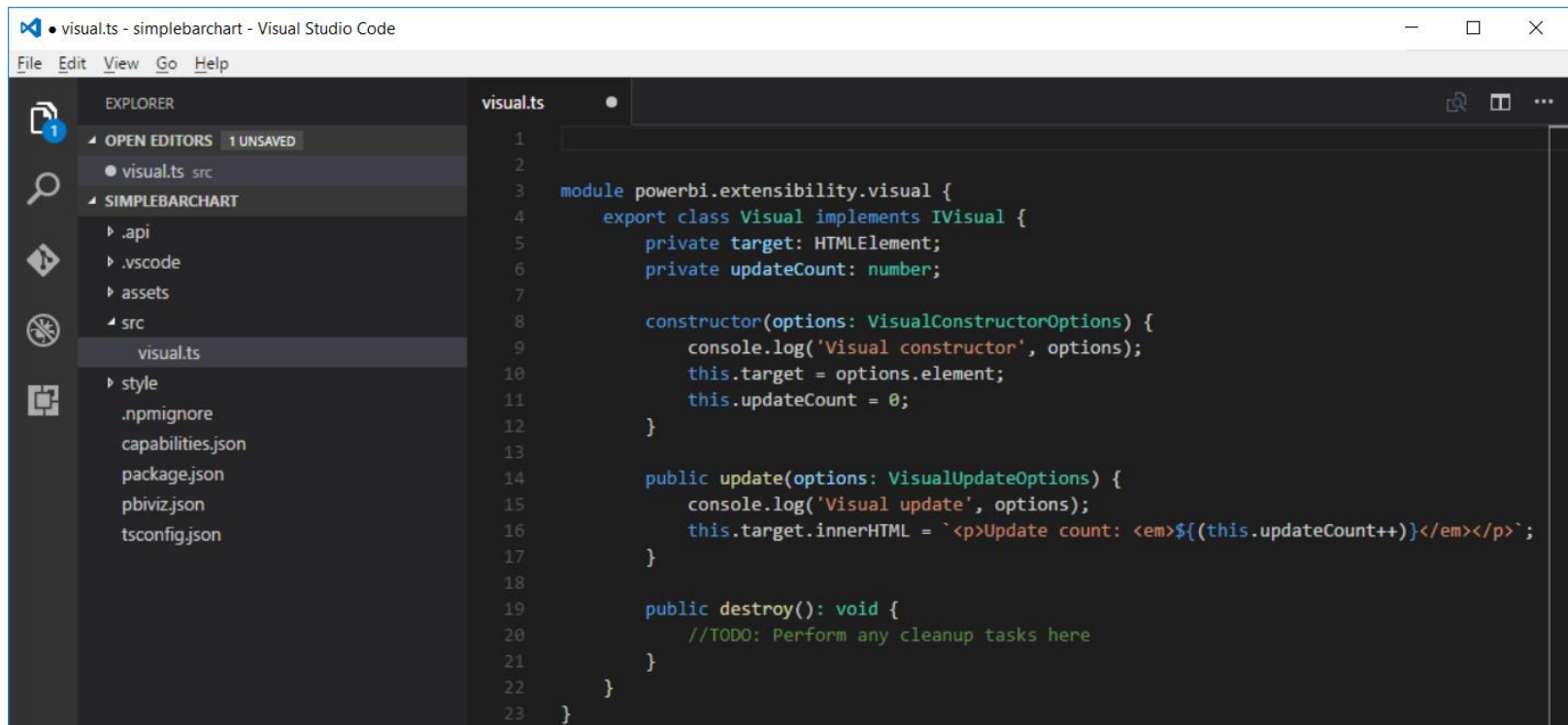
Install Visual Studio Code

- <http://code.visualstudio.com/>



Developing with Visual Studio Code

- Provides great development experience with node.js



```
1  
2  
3 module powerbi.extensibility.visual {  
4   export class Visual implements IVisual {  
5     private target: HTMLElement;  
6     private updateCount: number;  
7  
8     constructor(options: VisualConstructorOptions) {  
9       console.log('Visual constructor', options);  
10      this.target = options.element;  
11      this.updateCount = 0;  
12    }  
13  
14    public update(options: VisualUpdateOptions) {  
15      console.log('Visual update', options);  
16      this.target.innerHTML = `

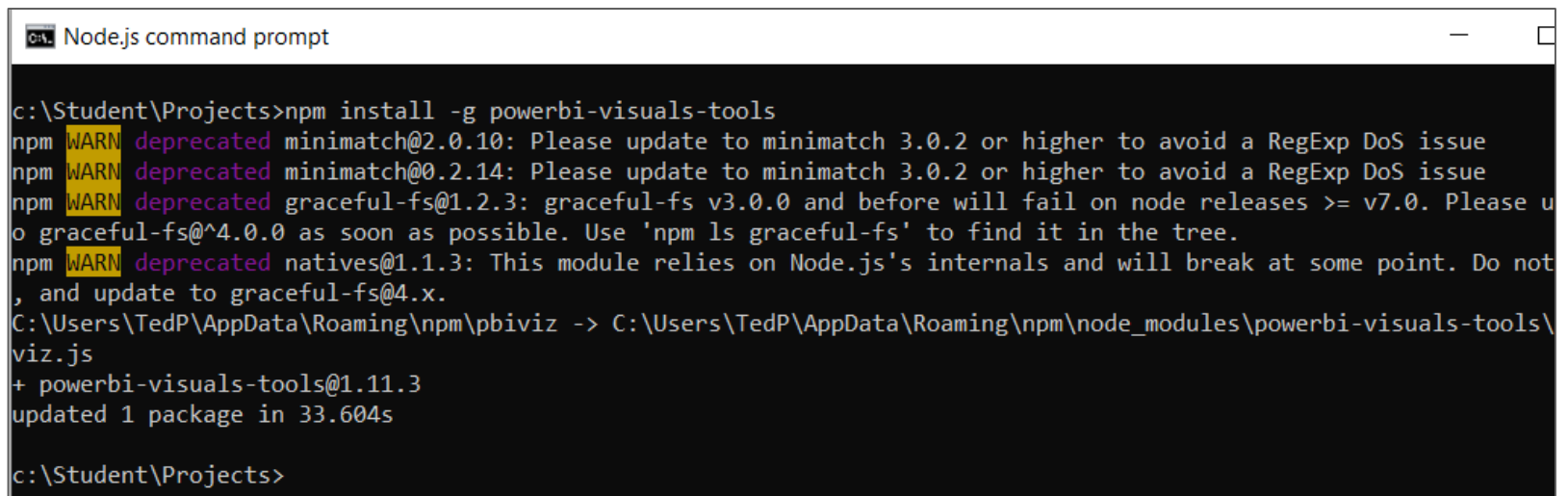
Update count: <em>${(this.updateCount++)}</em></p>`;  
17    }  
18  
19    public destroy(): void {  
20      //TODO: Perform any cleanup tasks here  
21    }  
22  }  
23 }


```



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



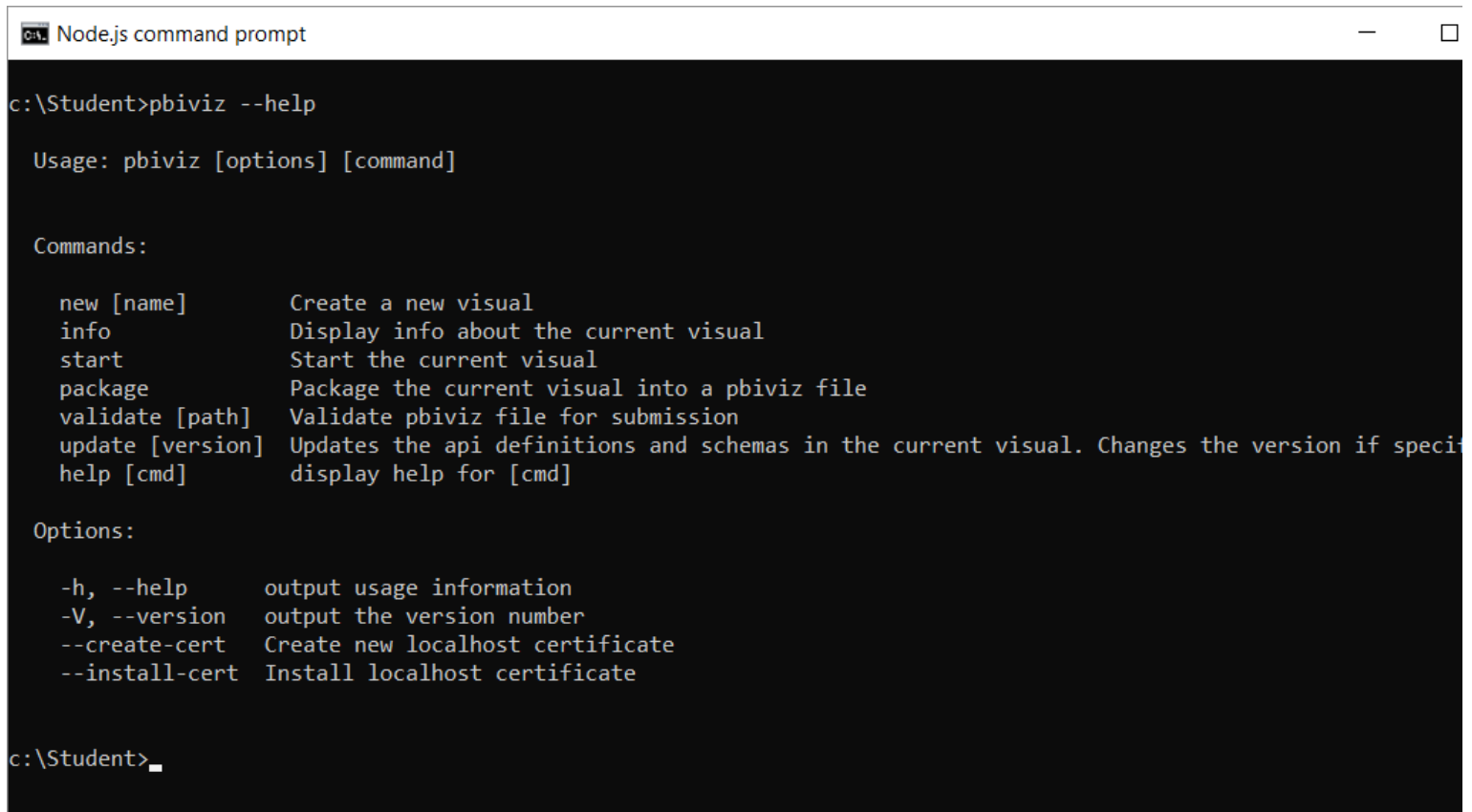
```
Node.js command prompt

c:\Student\Projects>npm install -g powerbi-visuals-tools
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Please u
o graceful-fs@^4.0.0 as soon as possible. Use 'npm ls graceful-fs' to find it in the tree.
npm WARN deprecated natives@1.1.3: This module relies on Node.js's internals and will break at some point. Do not
, and update to graceful-fs@4.x.
C:\Users\TedP\AppData\Roaming\npm\pbiviz -> C:\Users\TedP\AppData\Roaming\npm\node_modules\powerbi-visuals-tools\
viz.js
+ powerbi-visuals-tools@1.11.3
updated 1 package in 33.604s

c:\Student\Projects>
```

Getting Started with PBIVIZ

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



```
Node.js command prompt

c:\Student>pbiviz --help

Usage: pbiviz [options] [command]

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
  validate [path] Validate pbiviz file for submission
  update [version] Updates the api definitions and schemas in the current visual. Changes the version if specified
  help [cmd]      display help for [cmd]

Options:

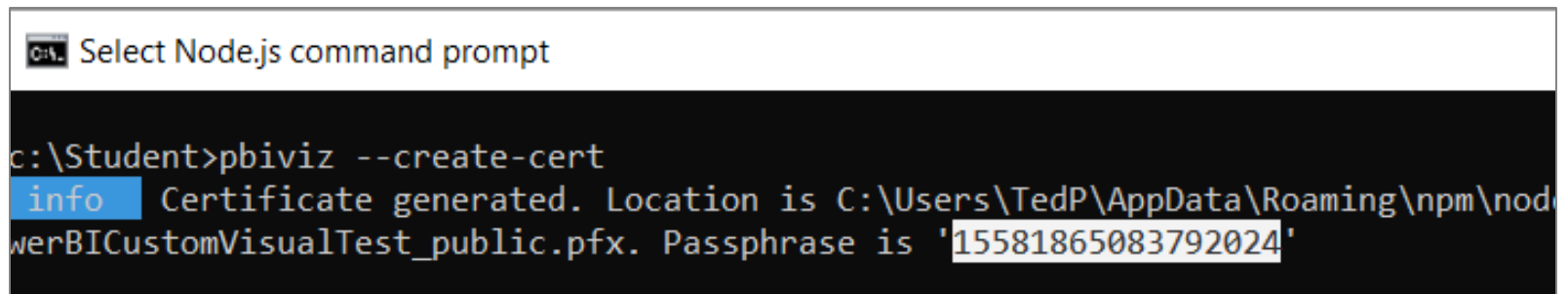
  -h, --help      output usage information
  -V, --version    output the version number
  --create-cert    Create new localhost certificate
  --install-cert   Install localhost certificate

c:\Student>
```



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 --create-cert** command
 - You must copy a passphrase to properly install the certificate



```
C:\> Select Node.js command prompt

c:\Student>pbiviz --create-cert
info Certificate generated. Location is C:\Users\TedP\AppData\Roaming\npm\nod
werBICustomVisualTest_public.pfx. Passphrase is '15581865083792024'
```



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

```
Node.js command prompt

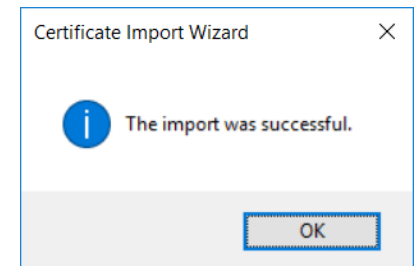
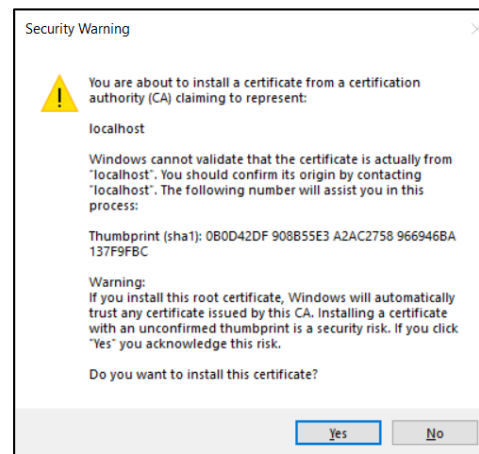
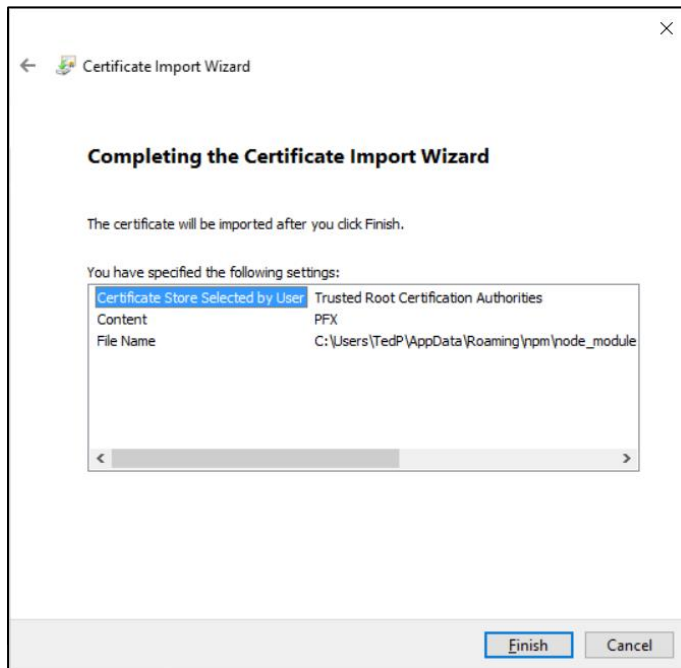
c:\Student>pbiviz --install-cert
info Use '15581865083792024' passphrase to install PFX certificate.

c:\Student>_
```



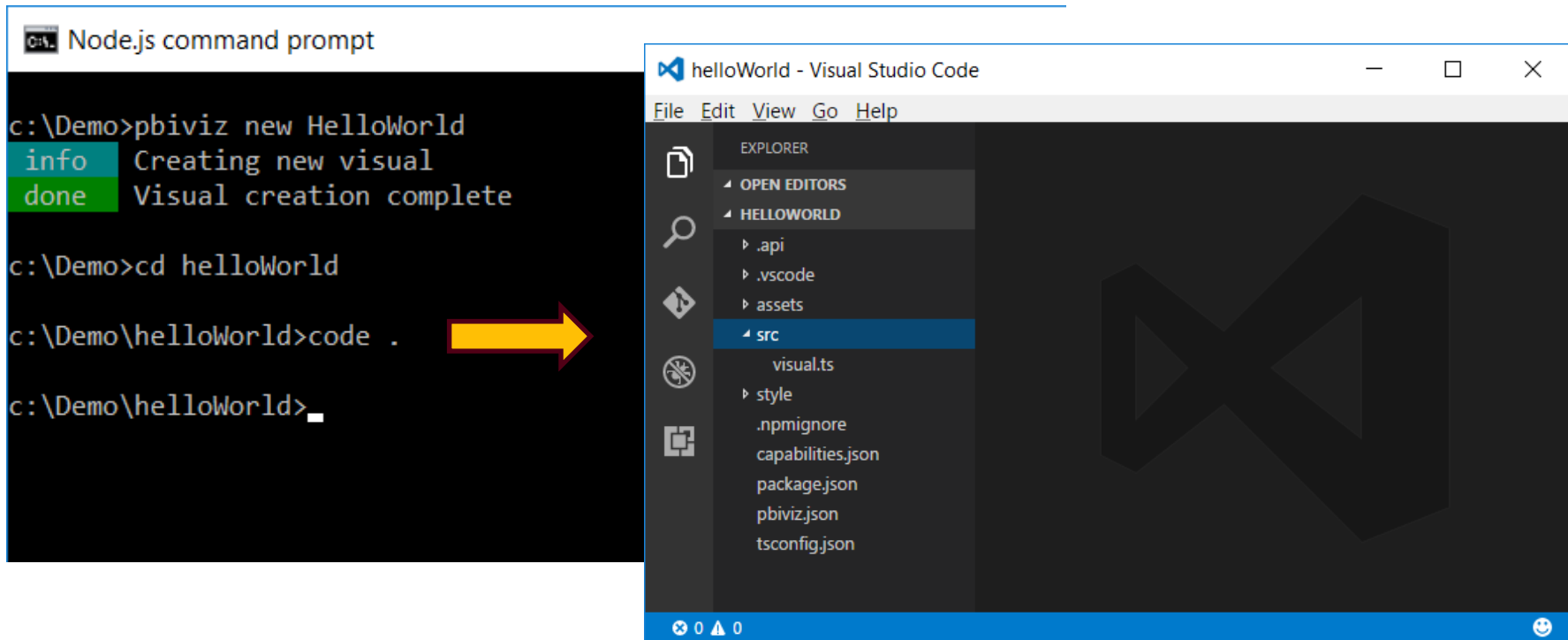
The Certificate Import Wizard

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



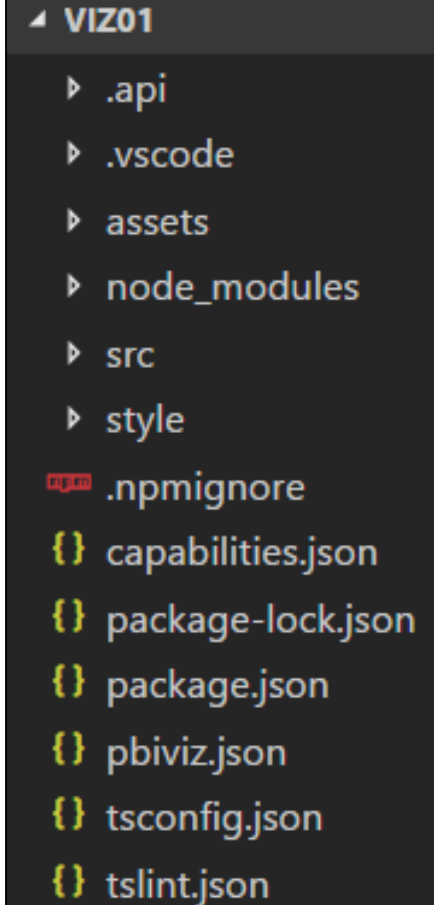
Creating a New Custom Visual Project

- Creating a new project
`pbiviz new <ProjectName>`
- Open the Project with Visual Studio Code
`code .`



Files in the new project

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

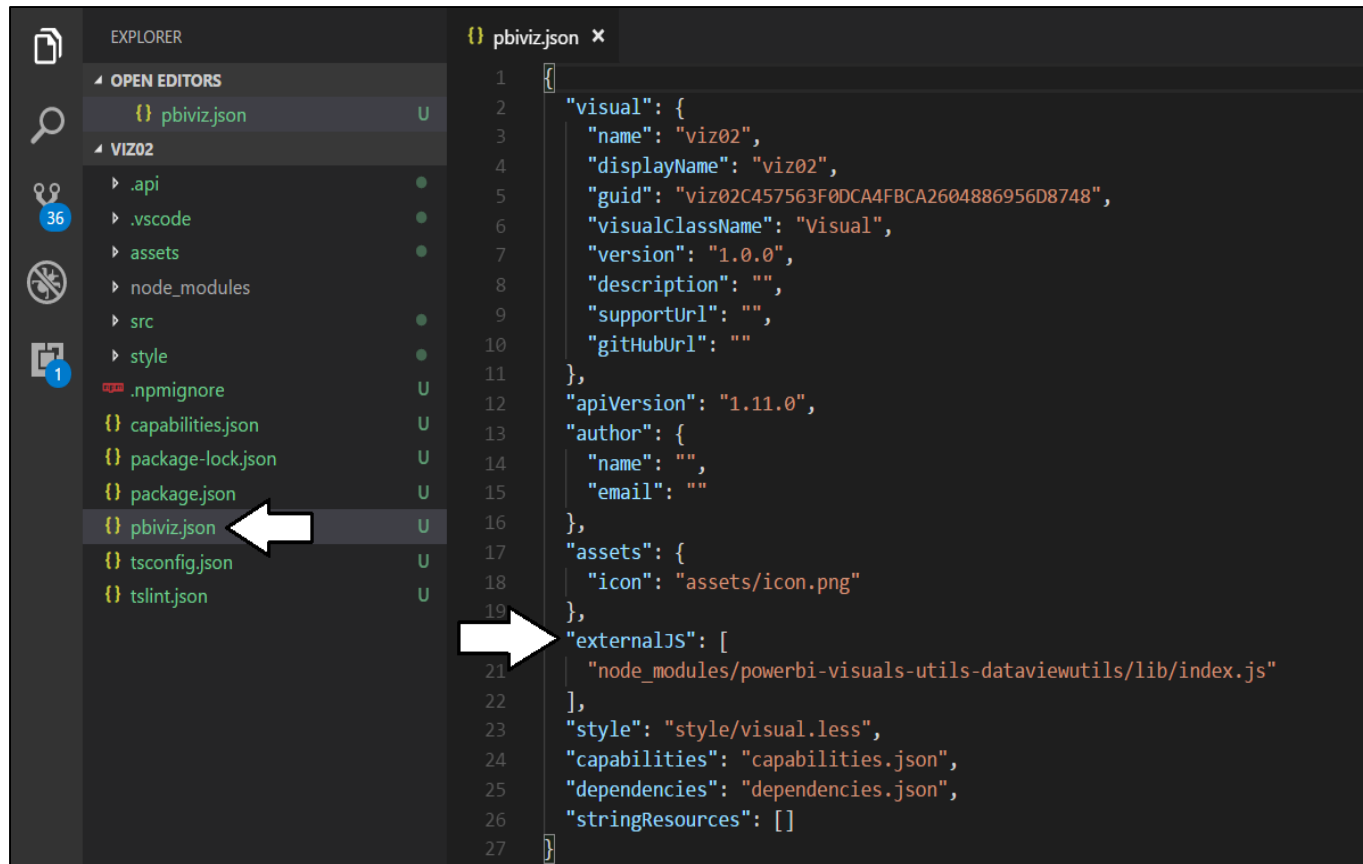


A screenshot of a file explorer window showing the project structure for a folder named 'VIZ01'. The folder is expanded, revealing several subfolders and files. The subfolders are: '.api', '.vscode', 'assets', 'node_modules', 'src', and 'style'. The files are: '.npmignore' (with a red 'new' tag), 'capabilities.json', 'package-lock.json', 'package.json', 'pbiviz.json', 'tsconfig.json', and 'tslint.json'. Each file is preceded by a yellow icon representing a JSON file.

```
▲ VIZ01
├── .api
├── .vscode
├── assets
├── node_modules
├── src
├── style
├── .npmignore
├── {} capabilities.json
├── {} package-lock.json
├── {} package.json
├── {} pbiviz.json
├── {} tsconfig.json
├── {} tslint.json
```

The pbiviz.json File

- Acts as top-level manifest file for custom visual project
 - External JS library files must be referenced in **externalJS** section



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the file structure of a project named 'VIZ02'. The file 'pbiviz.json' is highlighted with a white arrow pointing to it. The main editor area shows the content of 'pbiviz.json', which is a JSON manifest file. A white arrow points to the 'externalJS' array in the JSON, which contains a reference to a Power BI visual utility library.

```
1 {
2   "visual": {
3     "name": "viz02",
4     "displayName": "viz02",
5     "guid": "viz02C457563F0DCA4FBCA2604886956D8748",
6     "visualClassName": "Visual",
7     "version": "1.0.0",
8     "description": "",
9     "supportUrl": "",
10    "githubUrl": ""
11  },
12  "apiVersion": "1.11.0",
13  "author": {
14    "name": "",
15    "email": ""
16  },
17  "assets": {
18    "icon": "assets/icon.png"
19  },
20  "externalJS": [
21    "node_modules/powerbi-visuals-utils-dataviewutils/lib/index.js"
22  ],
23  "style": "style/visual.less",
24  "capabilities": "capabilities.json",
25  "dependencies": "dependencies.json",
26  "stringResources": []
27 }
```



Installing Support for jQuery

- Install package for jQuery library
`npm install jquery --save-dev`
- Install package for type definition files version 2.0.46
`npm install @types/jquery@2.0.46 --save-dev`
- Update **externalJS** section of **pbiviz.json**


```
"assets": {  
  "icon": "assets/icon.png"  
},  
"externalJS": [  
  "node_modules/powerbi-visuals-utils-dataviewutils/lib/index.js",  
  "node_modules/jquery/dist/jquery.js"  
],  
"style": "style/visual.less",
```



Installing Support for D3

- Install package for D3 library version 3
`npm install d3@3 --save-dev`
- Install package for type definition files version 3
`npm install @types/d3@3 --save-dev`
- Update **externalJS** section of **pbiviz.json**

```
17   "assets": {  
18     "icon": "assets/icon.png"  
19   },  
20   "externalJS": [  
21     "node_modules/powerbi-visuals-utils-dataviewutils/lib/index.js",  
22     "node_modules/d3/d3.js"  
23   ],  
24   "style": "style/visual.less",  
25   "capabilities": "capabilities.json",  
26   "dependencies": "dependencies.json",  
27   "stringResources": []  
28 }
```



The tsconfig.json File

- Used to add references to other TypeScript files
 - Controls which TypeScript files are passed to TypeScript compiler
 - No need to reference *.d.ts files in the **node_modules/@types** folder

```
{ } tsconfig.json •
1  {
2    "compilerOptions": {
3      "allowJs": true,
4      "emitDecoratorMetadata": true,
5      "experimentalDecorators": true,
6      "target": "ES5",
7      "sourceMap": true,
8      "out": "./.tmp/build/visual.js"
9    },
10   "files": [
11     ".api/v1.11.0/PowerBI-visuals.d.ts",
12     "node_modules/powerbi-visuals-utils-dataviewutils/lib/index.d.ts",
13     "node_modules/powerbi-visuals-utils-typeutils/lib/index.d.ts",
14     "node_modules/powerbi-visuals-utils-formattingutils/lib/index.d.ts",
15     "src/settings.ts",
16     "src/visual.ts"
17   ]
18 }
```



Authoring a Custom Visual Class

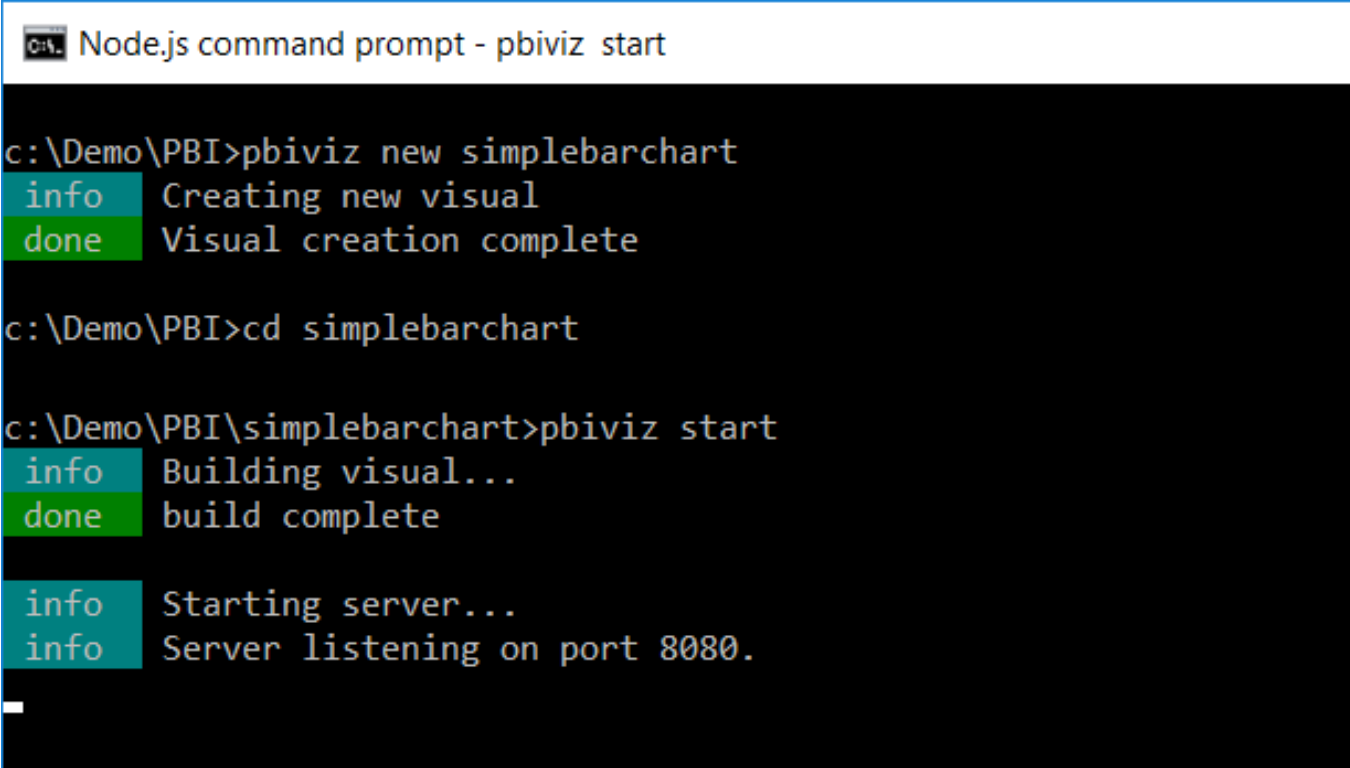
- Custom visual is a class that implements **IVisual**
 - Class must be defined in **powerbi.extensibility.visual** namespace
 - Minimum visual class must provide **update** method
 - Constructor and other lifecycle methods can be added

```
module powerbi.extensibility.visual {  
  
    export class Visual implements IVisual {  
  
        constructor(options: VisualConstructorOptions) {  
            // one-time initialization code  
        }  
  
        public update(options: VisualUpdateOptions) {  
            // called when viewport or data changes  
        }  
  
        public destroy(): void {  
            // add cleanup code here  
        }  
    }  
}
```



Running a Custom Visual Project

- Visual projects run & tested using **pbviz start** command
 - Command starts local debugging session in node.js.
 - Provides ability to run custom visual in the Power BI Service



```
Node.js command prompt - pbviz start

c:\Demo\PBI>pbviz new simplebarchart
info  Creating new visual
done  Visual creation complete

c:\Demo\PBI>cd simplebarchart

c:\Demo\PBI\simplebarchart>pbviz start
info  Building visual...
done  build complete

info  Starting server...
info  Server listening on port 8080.
```



Address In Use Error

- You can only start one session of PBIVIZ at a time
 - Session takes exclusive control of <https://localhost:8080>
 - Attempts to create secondary sessions will fail

```
PS C:\Student\CustomVisuals\betsy\betsy> pbiviz start
info Building visual...
done build complete

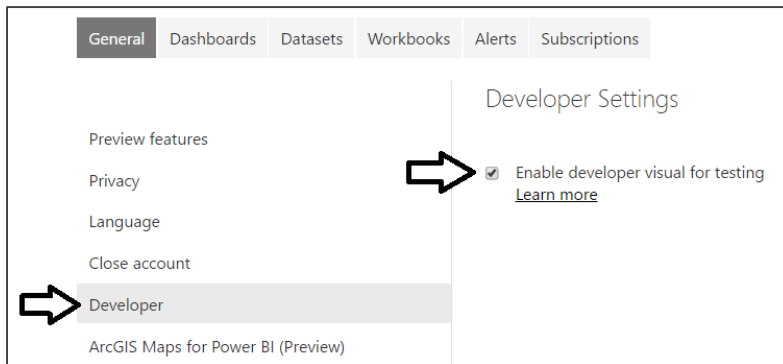
info Starting server...
events.js:183
    throw er; // Unhandled 'error' event
    ^

Error: listen EADDRINUSE :::8080
    at Object._errnoException (util.js:1022:11)
    at _exceptionWithHostPort (util.js:1044:20)
    at Server.setupListenHandle [as _listen2] (net.js:1367:14)
    at listenInCluster (net.js:1408:12)
    at Server.listen (net.js:1492:7)
    at Promise (C:\Users\TedP\AppData\Roaming\npm\node_modules\powerbi-visuals-tools\lib\VisualServer.js:96:64)
    at new Promise (<anonymous>)
    at VisualServer.start (C:\Users\TedP\AppData\Roaming\npm\node_modules\powerbi-visuals-tools\lib\VisualServer.js:59:16)
    at builder.startWatcher.then (C:\Users\TedP\AppData\Roaming\npm\node_modules\powerbi-visuals-tools\bin\pbiviz-start.js:77:20)
    at <anonymous>
PS C:\Student\CustomVisuals\betsy\betsy> |
```

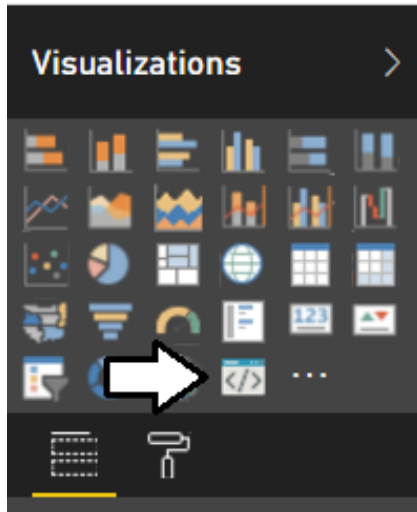


The Developer Visual

- Must be enabled on Developer Settings page

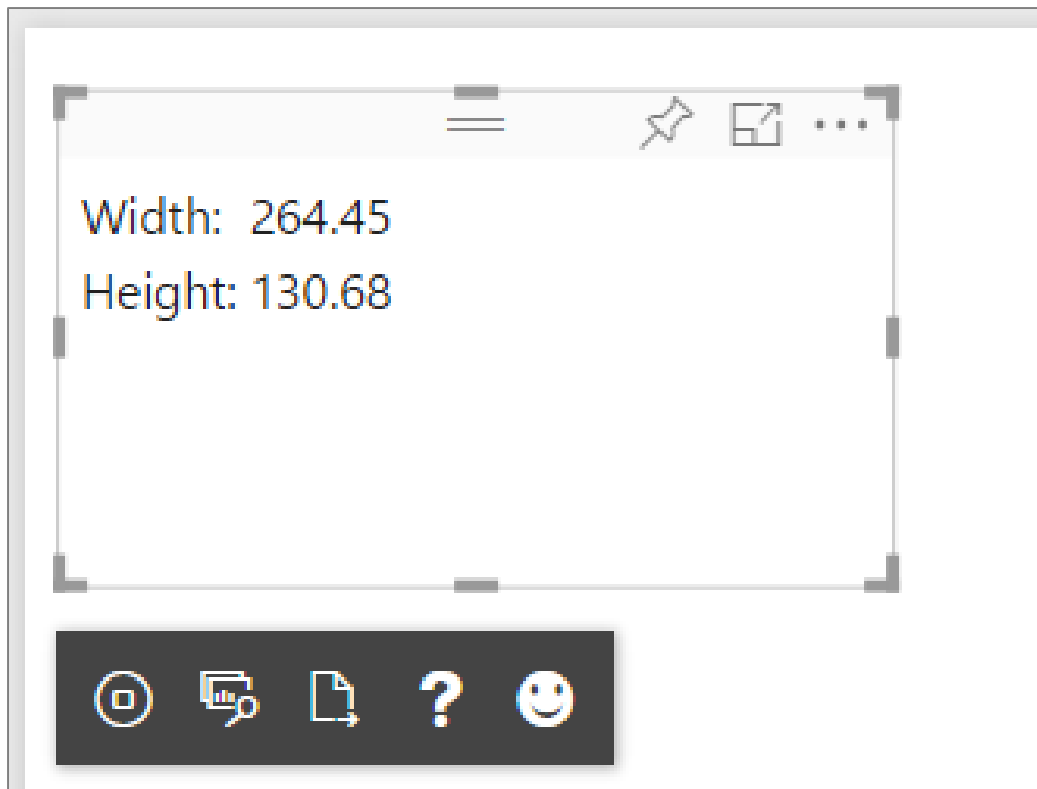


- Provides new visual for testing and debugging custom visuals



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



Summary

- ✓ Custom Visuals in Power BI
- ✓ Node.JS and the Cross-platform Toolchain
- ✓ Creating Projects with the PBIVIZ CLI
- ✓ Custom Visual Project Structure
- ✓ Adding Typed Definition Files
- ✓ Testing and Debugging a Custom Visual

