

Getting Started with Node.js



Agenda

- Introduction to Node.JS and NPM
- Automating Build Tasks using Gulp
- Bundling Project Assets using Webpack
- Developing with React.js, TypeScript and Webpack
- Developing with SharePoint Framework
- Developing Custom Visuals for Power BI





What is Node.js

- Node.js was created to develop server-side applications in JavaScript
 - Node.js was initially created by Ryan Dahl in 2009
 - Built using Google's V8 JavaScript engine
 - Created to solve Apache HTTP Server's inability to deal with concurrency
 - Node.js offers single-threaded, non-blocking, asynchronously programming
- JavaScript run-time environment based on Google V8 engine
 - JavaScript execution environment for web servers and development machines
 - It's free, cross-platform and open-source
 - Includes Node Package Manager (npm) and lots of available packages
- What are the primary motivations for using Node.js
 - Server-side development with web applications and web services
 - Development environment with package management
 - Development with SharePoint Framework (SPX) of Power BI custom visuals



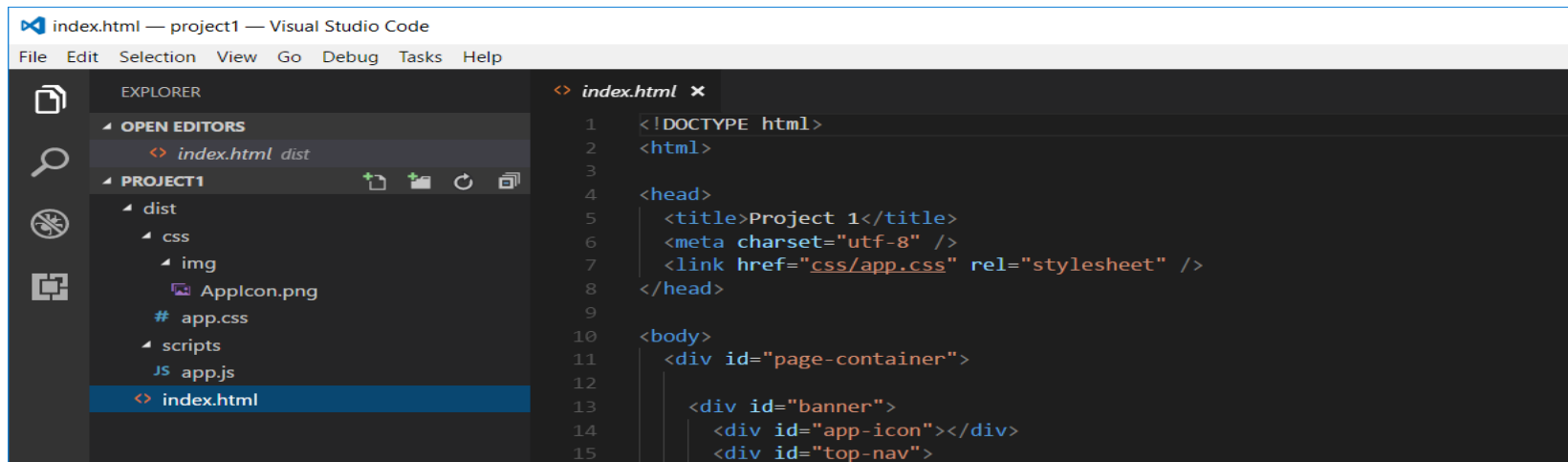
Installing node.js

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



Developing with Visual Studio Code

- Node.js is agnostic when it comes to developer IDE
 - There are many different IDEs that people use with Node.js
 - This course will be using Visual Studio Code

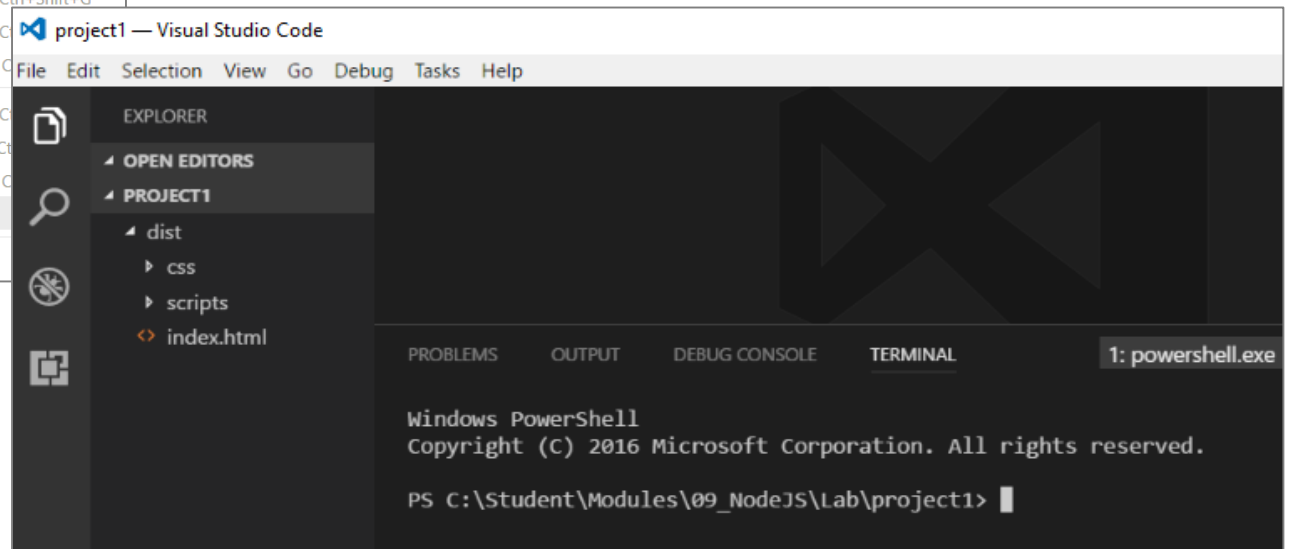
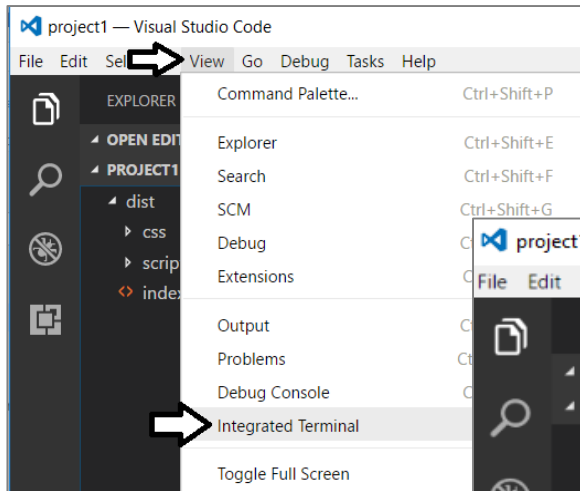


- Visual Studio is not a good fit for Node.js development
 - Visual Studio solution & project files incompatible with Node.js



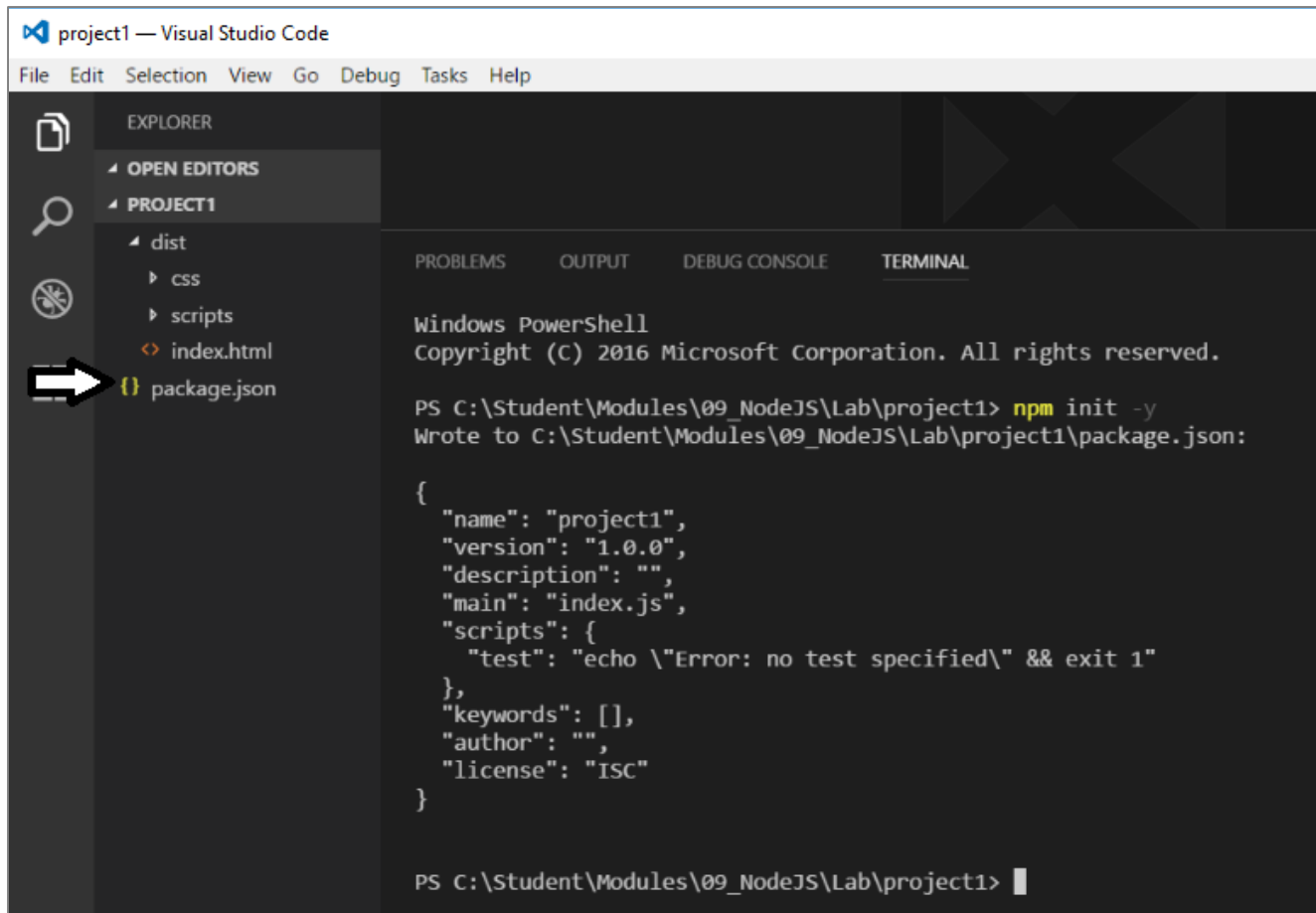
Integrated Terminal

- Use the Integrated Terminal to execute `npm` command



npm init

- Node.js projects initialized with `npm init` command
- This command created the **package.json** file



```
project1 — Visual Studio Code
File Edit Selection View Go Debug Tasks Help

EXPLORER
├─ OPEN EDITORS
├─ PROJECT1
│   ├─ dist
│   │   └─ css
│   │   └─ scripts
│   └─ index.html
└─ package.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

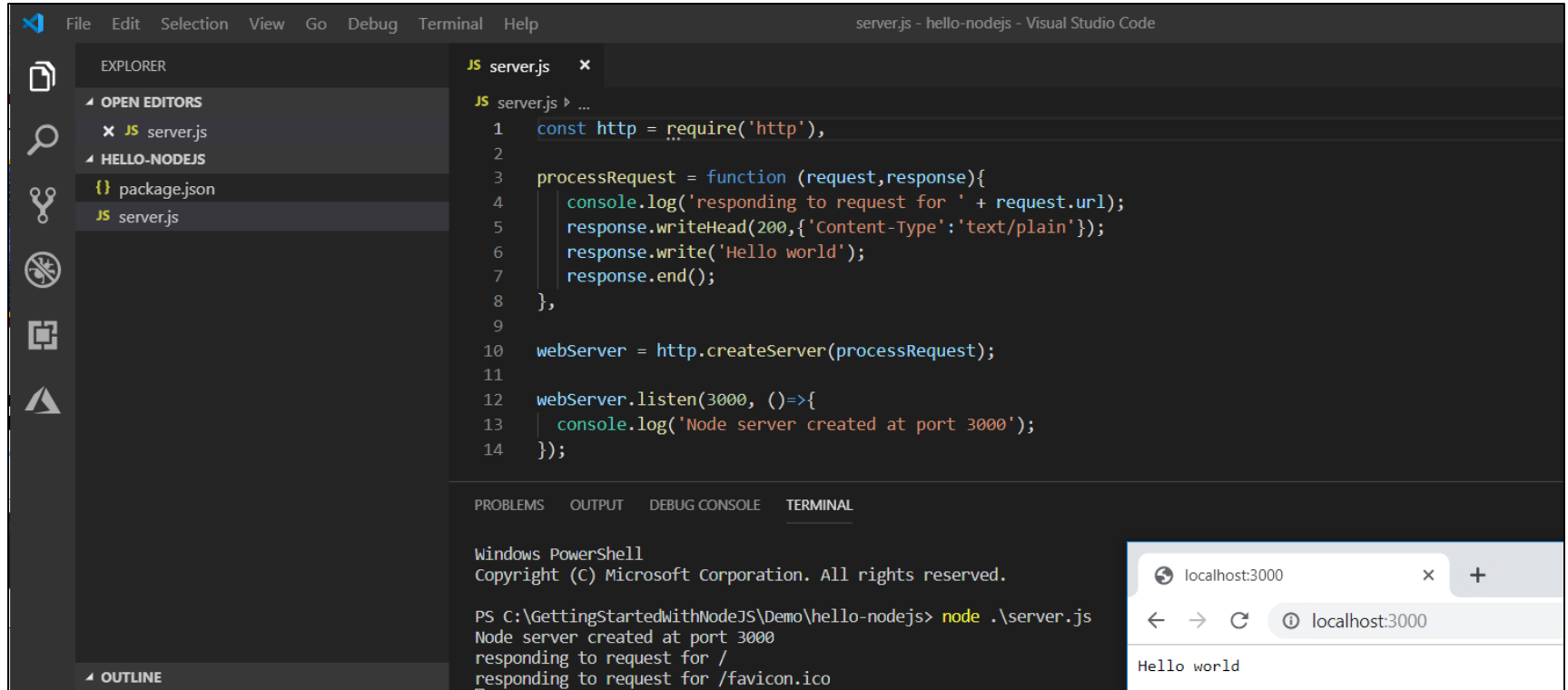
PS C:\Student\Modules\09_NodeJS\Lab\project1> npm init -y
Wrote to C:\Student\Modules\09_NodeJS\Lab\project1\package.json:

{
  "name": "project1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Student\Modules\09_NodeJS\Lab\project1>
```



Hello Node.js



The image shows a Visual Studio Code editor window titled "server.js - hello-nodejs - Visual Studio Code". The interface is divided into three main sections: Explorer, Editor, and Terminal/Output.

Explorer: The left sidebar shows the file explorer with "package.json" and "server.js" listed under the "HELLO-NODEJS" folder. "server.js" is currently selected.

Editor: The main editor area displays the content of "server.js". The code is as follows:

```
1  const http = require('http'),
2
3  processRequest = function (request,response){
4      console.log('responding to request for ' + request.url);
5      response.writeHead(200,{ 'Content-Type': 'text/plain' });
6      response.write('Hello world');
7      response.end();
8  },
9
10 webServer = http.createServer(processRequest);
11
12 webServer.listen(3000, ()=>{
13     console.log('Node server created at port 3000');
14 });
```

Terminal: The bottom section shows the Windows PowerShell terminal. It displays the command to run the server and its output:

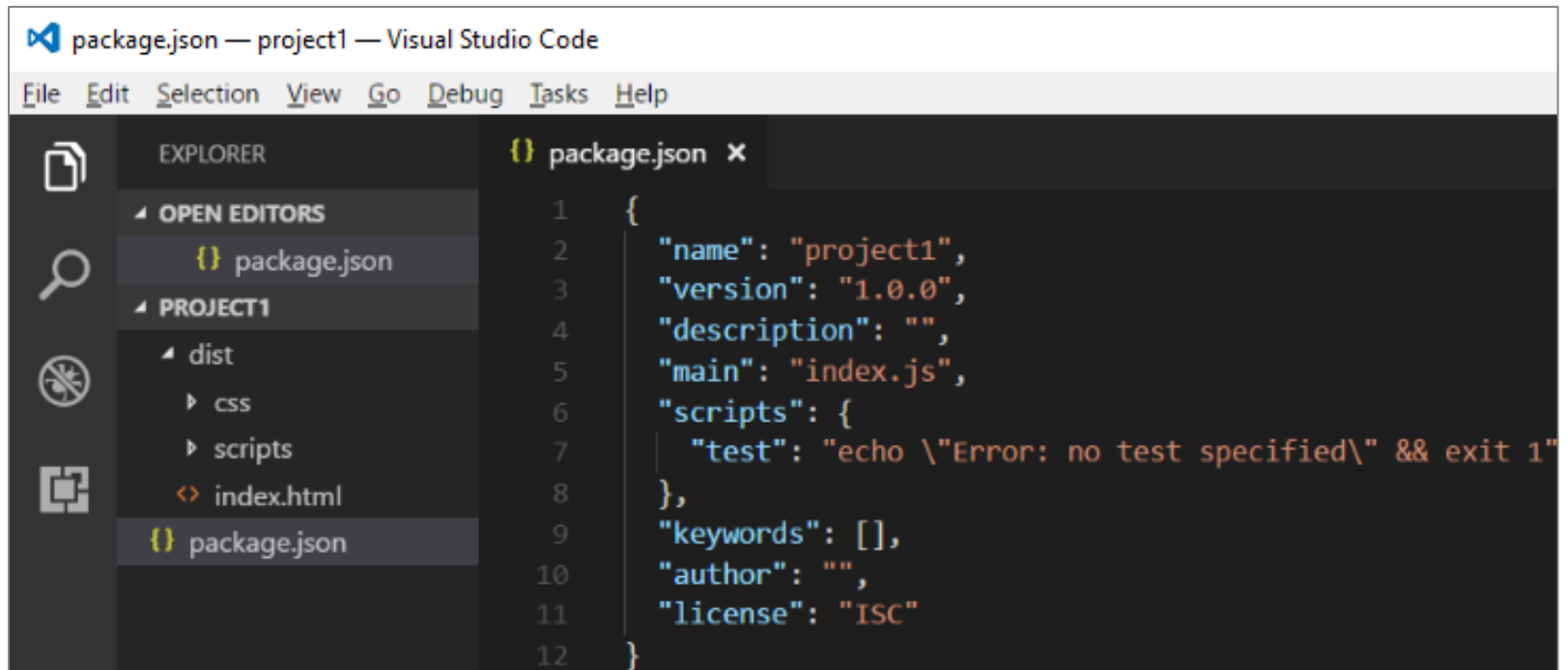
```
PS C:\GettingStartedWithNodeJS\Demo\hello-nodejs> node .\server.js
Node server created at port 3000
responding to request for /
responding to request for /favicon.ico
```

Browser: A web browser window is open in the bottom right corner, showing the address "localhost:3000". The page content is "Hello world".



package.json

- **package.json** serves as project manifest file
 - Tracks project name and version number
 - Tracks installed package dependencies



The screenshot shows the Visual Studio Code interface with a file named `package.json` open in the editor. The Explorer sidebar on the left shows the project structure, including a `dist` folder with `css` and `scripts` subfolders, and an `index.html` file. The `package.json` file is selected in the Explorer. The editor displays the following JSON content:

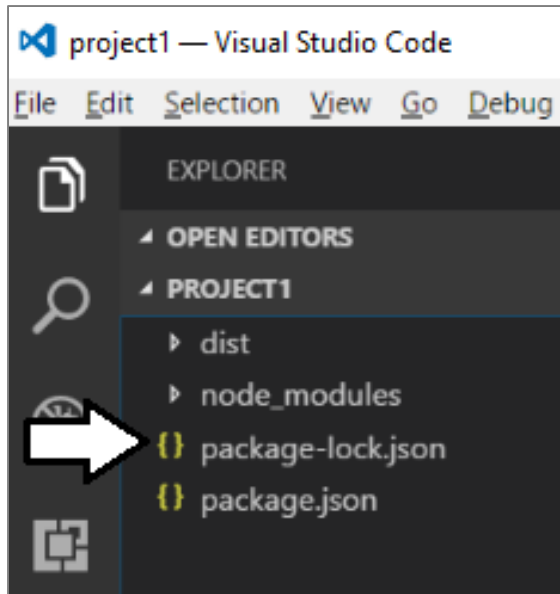

```
1 {
2   "name": "project1",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC"
12 }
```



Installing Packages

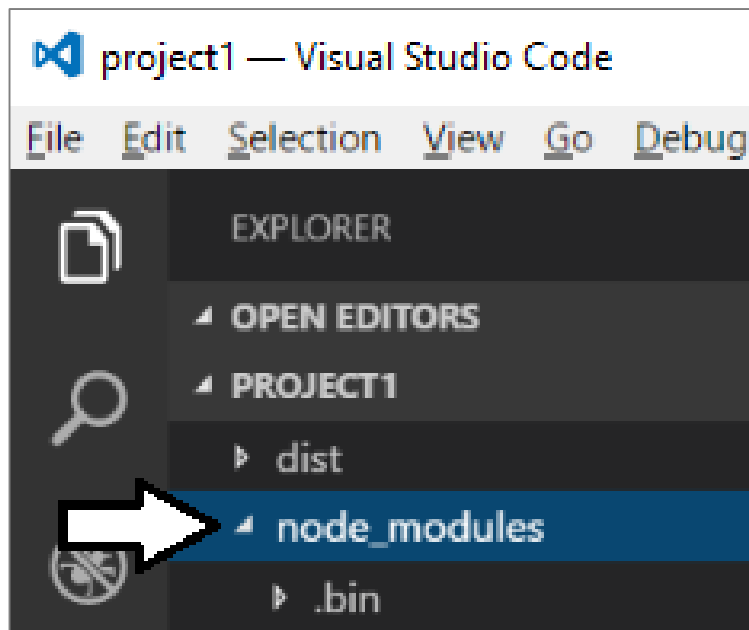
```
npm install browser-sync --save-dev
```

```
"devDependencies": {  
  "browser-sync": "^2.18.12"  
}
```



node_modules folder

- Package files copied into **node_modules** folder
 - This folder often contain 100s of packages for a project
 - Contents of folder not saved into source control
 - Contents can be restored with **npm install** command



Configuring a Server-side Web Server

- Node.js does not provide its own web server
 - Instead, you must install a npm package to provide web server
 - There are many different packages to choose from
- Example packages which provide a web server for testing
 - http-server
 - express
 - Browser-sync (*this is the one we will be using*)

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npm install browser-sync --save-dev
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN project1@1.0.0 No description
npm WARN project1@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
+ browser-sync@2.24.6
added 222 packages in 23.346s
PS C:\Student\Modules\02_NodeJS\Lab\project1> █
```



Using browser-sync to Serve Content

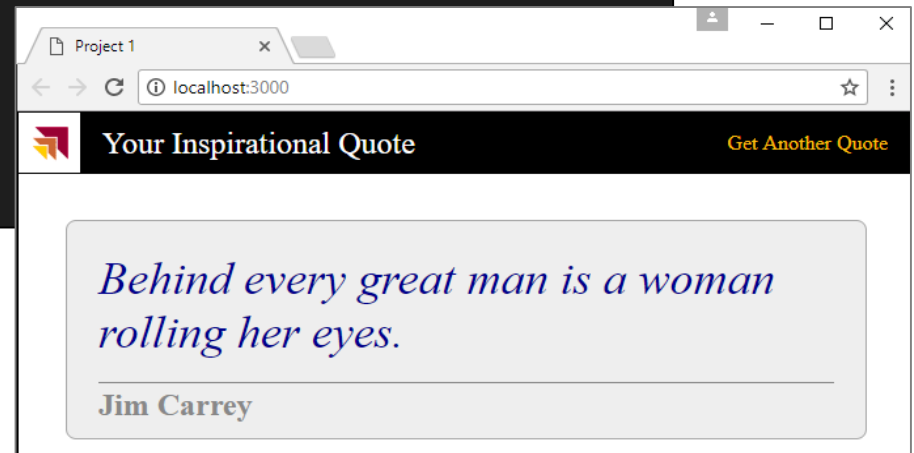
- **browser-sync start** command used to start web server
 - **--server** parameters references root folder with **index.html**

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> browser-sync start --server dist  
[Browsersync] Access URLs:
```

```
-----  
Local: http://localhost:3000  
External: http://10.0.0.3:3000  
-----
```

```
UI: http://localhost:3001  
UI External: http://10.0.0.3:3001  
-----
```

```
[Browsersync] Serving files from: dist
```



Stopping the Web Server Session

- Type CTRL + C into console to interrupt session

```
Local: http://localhost:3000
External: http://10.0.0.3:3000
-----
UI: http://localhost:3001
UI External: http://10.0.0.3:3001
-----
[Browsersync] Serving files from: dist
^CTerminate batch job (Y/N)? █
```



Installing the TypeScript Package

- typescript package must be installed into project
 - Installed just like any other npm package

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npm install typescript --save-dev
npm WARN project1@1.0.0 No description
npm WARN project1@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4
+ typescript@3.0.1
added 1 package in 8.156s
PS C:\Student\Modules\02_NodeJS\Lab\project1> █
```

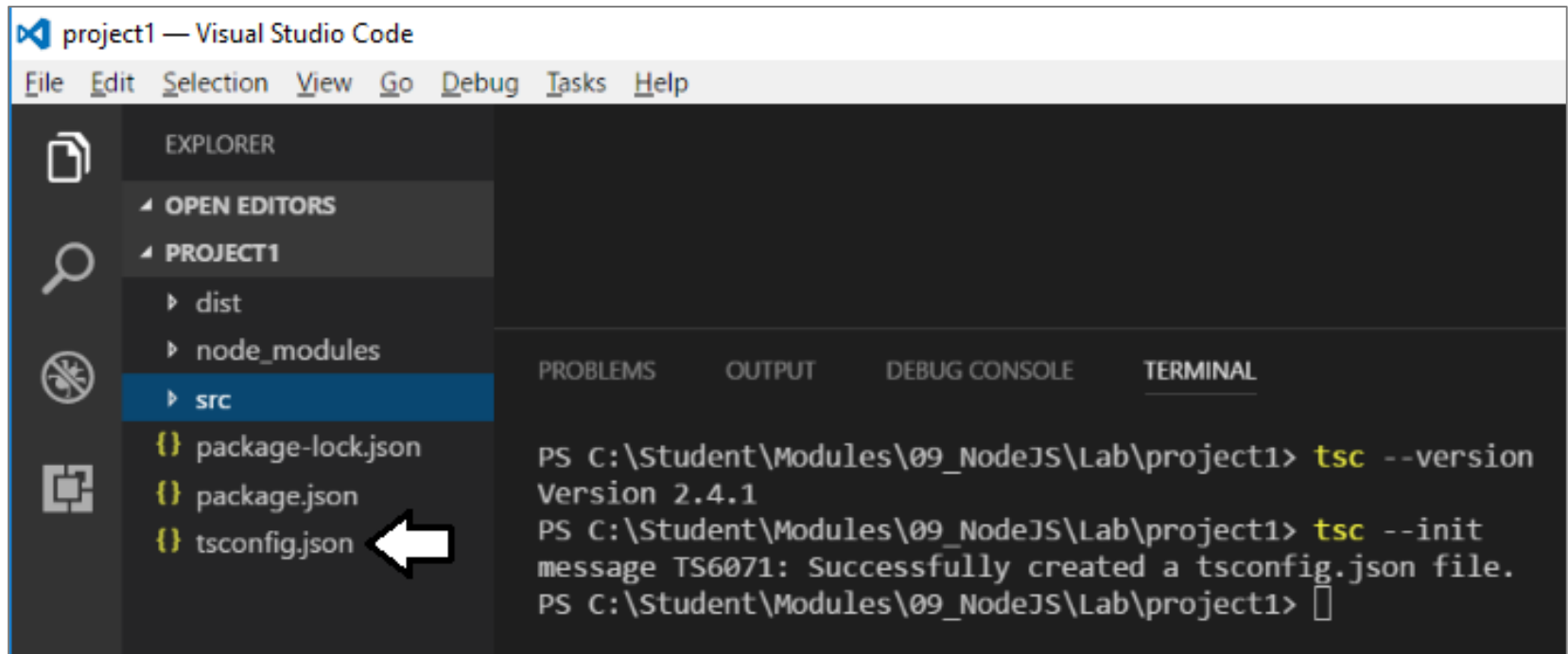
- Take note of version number of typescript package
 - typescript version may vary from one project to another
 - Determine project-specific version using **npx tsc --version**

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npx tsc --version
npx: installed 1 in 3.79s
Path must be a string. Received undefined
C:\Student\Modules\02_NodeJS\Lab\project1\node_modules\typescript\bin\tsc
Version 3.0.1
PS C:\Student\Modules\02_NodeJS\Lab\project1>
```



Generating tsconfig.json

- Typescript compilation controlled using **tsconfig.json** file
 - Generated using **tsc --init** command



The screenshot shows the Visual Studio Code interface for a project named 'project1'. The Explorer sidebar on the left displays the file structure: 'PROJECT1' contains 'dist', 'node_modules', and 'src'. Below these, the files 'package-lock.json', 'package.json', and 'tsconfig.json' are listed. A white arrow points to 'tsconfig.json'. The Terminal panel on the right shows the following commands and output:

```
PS C:\Student\Modules\09_NodeJS\Lab\project1> tsc --version
Version 2.4.1
PS C:\Student\Modules\09_NodeJS\Lab\project1> tsc --init
message TS6071: Successfully created a tsconfig.json file.
PS C:\Student\Modules\09_NodeJS\Lab\project1> 
```



tsconfig.json

- Example of a **tsconfig.json** file

```
{  
  "compilerOptions": {  
    "noImplicitAny": true,  
    "removeComments": true,  
    "preserveConstEnums": true,  
    "outFile": "./dist/scripts/app.js",  
    "sourceMap": true,  
    "lib": [  
      "dom",  
      "es6"  
    ]  
  },  
  "files": [  
    "./src/scripts/app.ts"  
  ],  
  "exclude": [  
    "node_modules"  
  ]  
}
```



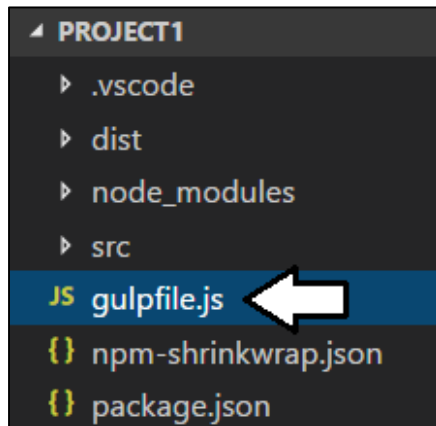
Agenda

- ✓ Introduction to Node.JS and NPM
- Automating Build Tasks using Gulp
 - Bundling Project Assets using Webpack
 - Developing with React.js, TypeScript and Webpack
 - Developing with SharePoint Framework
 - Developing Custom Visuals for Power BI



gulpfile.js

- Gulp tasks are programmed inside **gulpfile.js**
 - **Gulpfile.js** must be added to root of project



```
JS gulpfile.js x
var gulp = require('gulp');
var clean = require('gulp-clean');
var ts = require("gulp-typescript");
var tsProject = ts.createProject("tsconfig.json");
var sourcemaps = require('gulp-sourcemaps');
var browserSync = require('browser-sync');

gulp.task('clean', function () {
  console.log("Running clean task");
  return gulp.src('dist/', { read: false })
    .pipe(clean());
});

gulp.task('build', ['clean'], function () {
  console.log("Running build task");

  gulp.src('src/**/*.html').pipe(gulp.dest('dist'));
  gulp.src('src/css/**/*.css').pipe(gulp.dest('dist/css'));
  gulp.src('src/css/img/**/*.png').pipe(gulp.dest('dist/css/img'));

  return tsProject.src()
    .pipe(sourcemaps.init())
    .pipe(tsProject())
    .pipe(sourcemaps.write('.', { sourceRoot: './', includeContent: false }))
    .pipe(gulp.dest("."));
});
```

Gulp as a Task Runner

- Gulp serves as a Task Runner
 - Compiles TypeScript files to JavaScript
 - Compiles SASS files to CSS
 - Bundles and minifies JavaScript and CSS files

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npm install gulp --save-dev
npm WARN using --force I sure hope you know what you are doing.
npm WARN deprecated gulp-util@3.0.8: gulp-util is deprecated - replace it,
npm WARN deprecated graceful-fs@3.0.11: please upgrade to graceful-fs 4 for
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or
npm WARN deprecated graceful-fs@1.2.3: please upgrade to graceful-fs 4 for
> fsevents@1.2.4 install C:\Student\Modules\02_NodeJS\Lab\project1\node_mo
> node install

npm WARN project1@1.0.0 No description
npm WARN project1@1.0.0 No repository field.

+ gulp@3.9.1
added 75 packages in 6.404s
PS C:\Student\Modules\02_NodeJS\Lab\project1> █
```



Agenda

- ✓ Introduction to Node.JS and NPM
- ✓ Automating Build Tasks using Gulp
- Bundling Project Assets using Webpack
 - Developing with React.js, TypeScript and Webpack
 - Developing with SharePoint Framework
 - Developing Custom Visuals for Power BI



WebPack

- WebPack serves as a bundling utility
 - Bundles many js/ts files into a single file
 - Can handle dynamic module loading
 - Provides a dev server for testing and debugging
- When using Webpack 4
 - Install packages for webpack and webpack-cli
`npm install webpack webpack-cli --save-dev`



Dynamic Module Loading

- Webpack controls dynamic module loading
 - Your project just references app.ts
 - Compiler dynamically determines other files to include

```
TS app.ts x
import { Quote } from './quote';
import { QuoteManager } from './quote-manager';

$( () => {

  var displayNewQuote = (): void => {
    var quote: Quote = QuoteManager.getQuote();
    $("#quote").text(quote.value);
    $("#author").text(quote.author);
  }
});
```

```
TS quote.ts •
1  export class Quote {
2    value: string;
3    author: string;
4    constructor(value: string, author: string){
5      this.value = value;
6      this.author = author;
7    }
8  }
```

```
TS quote-manager.ts x
1  import { Quote } from './quote';
2
3  export class QuoteManager {
4
5    private static quotes: Quote[] = [
6      new Quote("Always borrow money from a partner", "John D. Rockefeller"),
7      new Quote("Behind every great man is a woman", "William Somerset Maugham"),
8      new Quote("In Hollywood a marriage is a business", "Marilyn Monroe")
9    ];
10
11    public static getQuote(): Quote {
12      return this.quotes[Math.floor(Math.random() * this.quotes.length)];
13    }
14  }
```



Webpack Loaders

- Loaders do two things
 - Identify which file or files should be transformed
 - Transform files and add them to dependency graph
- Example loaders
 - awesome-typescript-loader
 - style-loader
 - css-loader
 - url-loader



Webpack Plugins

- Webpack supports plugins in addition to loaders
 - commonly used to perform actions and custom functionality
 - Plugins act upon compilations or chunks of your bundled modules
- Examples Plugins
 - clean-webpack-plugin
 - copy-webpack-plugin
 - html-webpack-plugin



webpack.config.js

- Build process controlled through webpack.config.js

```
webpack.config.js •
const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');
const CopyWebpackPlugin = require('copy-webpack-plugin');
const CleanWebpackPlugin = require('clean-webpack-plugin')

module.exports = {
  entry: './src/scripts/app.ts',
  output: {
    filename: 'scripts/bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  resolve: {
    extensions: ['.js', '.ts']
  },
  plugins: [
    new CleanWebpackPlugin(['dist']),
    new HtmlWebpackPlugin({ template: path.join(__dirname, 'src', 'index.html') }),
    new CopyWebpackPlugin([{ from: './src/favicon.ico', to: 'favicon.ico' }])
  ],
  module: {
    rules: [
      { test: /\.ts$/, loader: 'awesome-typescript-loader' },
      { test: /\.css$/, use: ['style-loader', 'css-loader'] },
      { test: /\.(png|jpg|gif)$/, use: [{ loader: 'url-loader', options: { limit: 8192 } }] }
    ],
  },
  mode: "development",
  devtool: 'source-map'
};
```



Webpack Builds

- Running build process generates files for distribution

```
PS C:\Student\Modules\02_NodeJS\Lab\project2> npm run build
> project2@1.0.0 build C:\Student\Modules\02_NodeJS\Lab\project2
> webpack

clean-webpack-plugin: C:\Student\Modules\02_NodeJS\Lab\project2\dist has been removed.
i |atl|: Using typescript@3.0.1 from typescript
i |atl|: Using tsconfig.json from C:/Student/Modules/02_NodeJS/Lab/project2/tsconfig.json
i |atl|: Checking started in a separate process...
i |atl|: Time: 595ms
Hash: 9bd924fdc1391178039d
Version: webpack 4.16.4
Time: 5486ms
Built at: 2018-08-02 16:29:28

```

Asset	Size	Chunks		Chunk Names
scripts/bundle.js	839 KiB	main	[emitted]	main
index.html	714 bytes		[emitted]	
favicon.ico	1.12 KiB		[emitted]	

```
Entrypoint main = scripts/bundle.js
[./node_modules/css-loader/index.js!./src/css/app.css] ./node_modules/css-loader!./src/css/app.css 1.89 KiB {main} [built]
[./src/css/app.css] 1.05 KiB {main} [built]
[./src/css/img/AppIcon.png] 981 bytes {main} [built]
[./src/scripts/app.ts] 505 bytes {main} [built]
[./src/scripts/quote-manager.ts] 2.38 KiB {main} [built]
[./src/scripts/quote.ts] 275 bytes {main} [built]
+ 5 hidden modules
Child html-webpack-plugin for "index.html":
  1 asset
  Entrypoint undefined = index.html
[./node_modules/html-webpack-plugin/lib/loader.js!./src/index.html] 880 bytes {0} [built]
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 509 bytes {0} [built]
[./node_modules/webpack/buildin/module.js] (webpack)/buildin/module.js 519 bytes {0} [built]
+ 1 hidden module
PS C:\Student\Modules\02_NodeJS\Lab\project2> █
```

Webpack Dev Server

- Webpack provides its own development server
 - Install the webpack dev server package
`npm install webpack-dev-server --save-dev`
 - Run your project using the webpack dev server CLI
`webpack-dev-server --open`



Agenda

- ✓ Introduction to Node.JS and NPM
- ✓ Automating Build Tasks using Gulp
- ✓ Bundling Project Assets using Webpack
- Developing with React.js, TypeScript and Webpack
 - Developing with SharePoint Framework
 - Developing Custom Visuals for Power BI



Introducing React

- React is a library for building user interfaces
 - Not as all-encompassing as a framework like Angular
 - Focused on building HTML-based user experiences
 - Based on reusable component-based architecture
 - Components *react* to state changes by updating UI
 - React uses shadow DOM for efficient event handling
- React was originally designed for Facebook
 - Also a good fit for building SPFx web parts



Understanding JSX (and TSX)

- JSX provides better syntax for HTML composition
 - JSX allows extends JavaScript with XML-like syntax
 - JSX syntax must be transpiled into JavaScript code

```
var myHtml = <div id="myAppContainer" style={{ backgroundColor:"yellow", padding:8 }}>
    <h2>Hello JSX</h2>
    <p>I'm composing HTML elements using JSX syntax.</p>
</div>;

ReactDOM.render( myHtml , document.getElementById("app") );
```

- JSX/TSX is separate from React library
 - JSX/TSX commonly used in React development
 - Babel compiler used to transpile JSX to JavaScript
 - TypeScript compiler used to transpile TSX to JavaScript



Defining React Components using TypeScript

- Component is class extending `React.Component`
 - Component usually defined in its own **tsx** file
 - Component class must define **render** method

```
my-component.tsx •  
  
import * as React from 'react';  
  
export class MyComponent extends React.Component<any, any> {  
  render() {  
    return <h2>Hello from my component</h2>;  
  }  
}
```

- Component can be instantiated with JSX/TSX syntax

```
app.tsx •  
  
import * as ReactDOM from 'react-dom';  
  
import { MyComponent } from './components/my-component'  
  
window.onload = () => {  
  // Create and render component  
  ReactDOM.render( <MyComponent/>, document.getElementById("app") );  
}
```



Component Properties and State

- Component can contain properties and state
 - Properties are initialized by external components
 - Properties are read-only to hosting component
 - State is set internally by hosting component
 - Changing state triggers UI refresh by calling render
 - UI experience created by ***reacting*** to changes in state



React Component Properties

- Defining component with a property

```
component1.tsx •  
  
import * as React from 'react';  
  
export interface MyCustomProps {  
  Name: string;  
}  
  
export class Component1 extends React.Component<MyCustomProps, {}> {  
  render() {  
    return <div>Hello, my name is {this.props.Name}</div>;  
  }  
}
```

- Instantiating component with a property

```
ReactDOM.render(  
  <Component1 Name="Fred" />,  
  document.getElementById("app")  
)
```



Stateful Component

TS IBeanCounterProps.ts •

```
export interface IBeanCounterProps {  
  StartingValue: number;  
}
```

TS IBeanCounterState.ts •

```
export interface IBeanCounterState {  
  count: number;  
}
```

BeanCounter.tsx •

```
import * as React from 'react';  
import styles from './BeanCounter.module.scss';  
import { IBeanCounterProps } from './IBeanCounterProps';  
import { IBeanCounterState } from './IBeanCounterState';  
  
export default class BeanCounter extends React.Component<IBeanCounterProps, IBeanCounterState> {  
  constructor(props: any) {  
    super(props);  
    this.state = { count: this.props.StartingValue };  
  }  
  
  private incrementCounter() {  
    var previousCount: number = this.state.count;  
    this.setState({ count: previousCount + 1 });  
  }  
}
```



Stateful Component Rendering

BeanCounter.tsx

```
import * as React from 'react';
import styles from './BeanCounter.module.scss';
import { IBeanCounterProps } from './IBeanCounterProps';
import { IBeanCounterState } from './IBeanCounterState';

export default class BeanCounter extends React.Component<IBeanCounterProps, IBeanCounterState> {

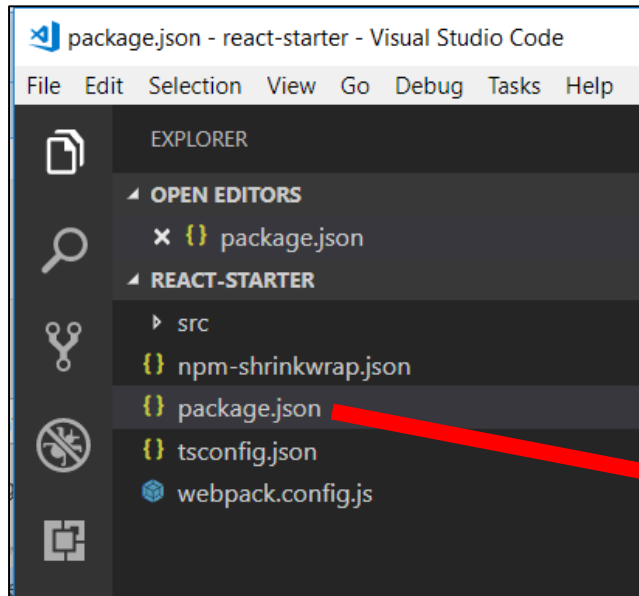
  constructor(props: any) {
    super(props);
    this.state = { count: this.props.StartingValue };
  }

  private incrementCounter() {
    var previousCount: number = this.state.count;
    this.setState({ count: previousCount + 1 });
  }

  public render(): React.ReactElement<IBeanCounterProps> {
    return (
      <div className={styles.beanCounter}>
        <h3>Mr Bean Counter</h3>
        <div className={styles.toolbar}>
          <button onClick={(event) => { this.incrementCounter(); }} >Add another Bean</button>
        </div>
        <div className={styles.beanCounterDisplay} >
          Bean Count: {this.state.count}
        </div>
      </div>
    );
  }
}
```

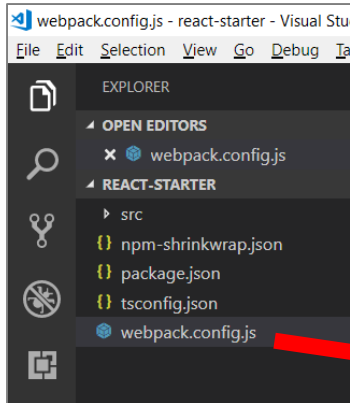


Starter Project - package.json



```
{ package.json •
{
  "name": "react-starter",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build": "webpack",
    "start": "webpack-dev-server --open --history-api-fallback"
  },
  "devDependencies": {
    "@types/react": "^16.4.13",
    "@types/react-dom": "^16.0.7",
    "awesome-typescript-loader": "^5.2.0",
    "bootstrap": "^4.1.3",
    "clean-webpack-plugin": "^0.1.19",
    "copy-webpack-plugin": "^4.5.2",
    "css-loader": "^0.28.11",
    "expose-loader": "^0.7.5",
    "file-loader": "^1.1.11",
    "html-webpack-plugin": "^3.2.0",
    "jquery": "3.3.1",
    "popper.js": "1.14.4",
    "react": "^16.4.2",
    "react-dom": "^16.4.2",
    "style-loader": "^0.21.0",
    "typescript": "3.0.1",
    "url-loader": "1.0.1",
    "webpack": "4.16.4",
    "webpack-cli": "3.1.0",
    "webpack-dev-server": "3.1.5"
  }
}
```

Starter Project - webpack.config.js



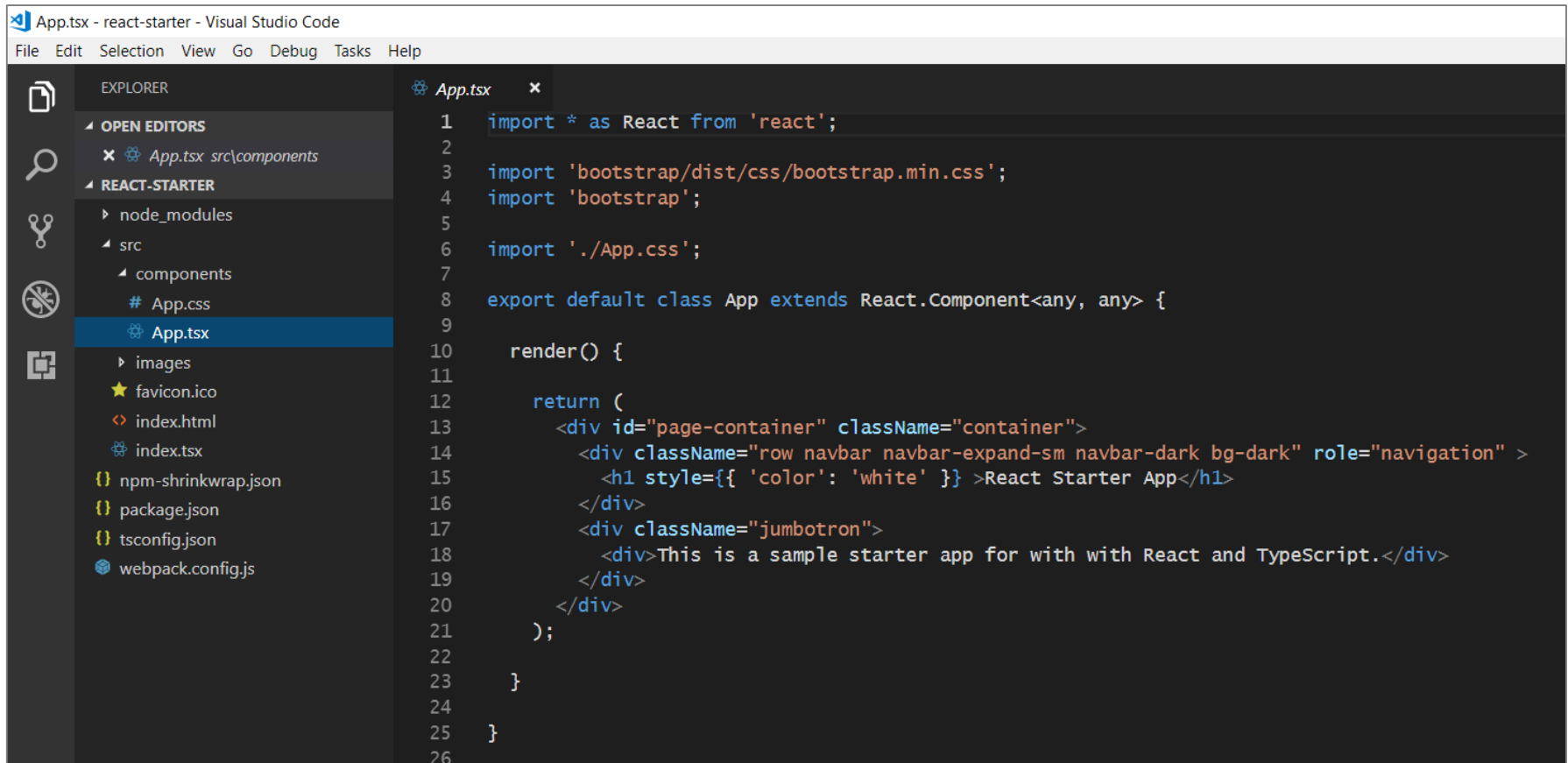
```
webpack.config.js x
const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');
const CopyWebpackPlugin = require('copy-webpack-plugin');
const CleanWebpackPlugin = require('clean-webpack-plugin')

module.exports = {
  entry: './src/index.tsx',
  output: {
    filename: 'scripts/bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  resolve: {
    extensions: ['.js', '.json', '.ts', '.tsx'],
  },
  plugins: [
    new CleanWebpackPlugin(['dist']),
    new HtmlWebpackPlugin({ template: path.join(__dirname, 'src', 'index.html') }),
    new CopyWebpackPlugin([{ from: './src/favicon.ico', to: 'favicon.ico' }])
  ],
  module: {
    rules: [
      { test: /\.ts|tsx$/, loader: 'awesome-typescript-loader' },
      { test: /\.css$/, use: ['style-loader', 'css-loader'] },
      { test: /\.(png|jpg|gif)$/, use: [{ loader: 'url-loader', options: { limit: 8192 } }] }
    ],
  },
  mode: "development",
  devtool: 'source-map',
  devtool: 'cheap-eval-source-map'
};
```



The Top-level App Component



App.tsx - react-starter - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

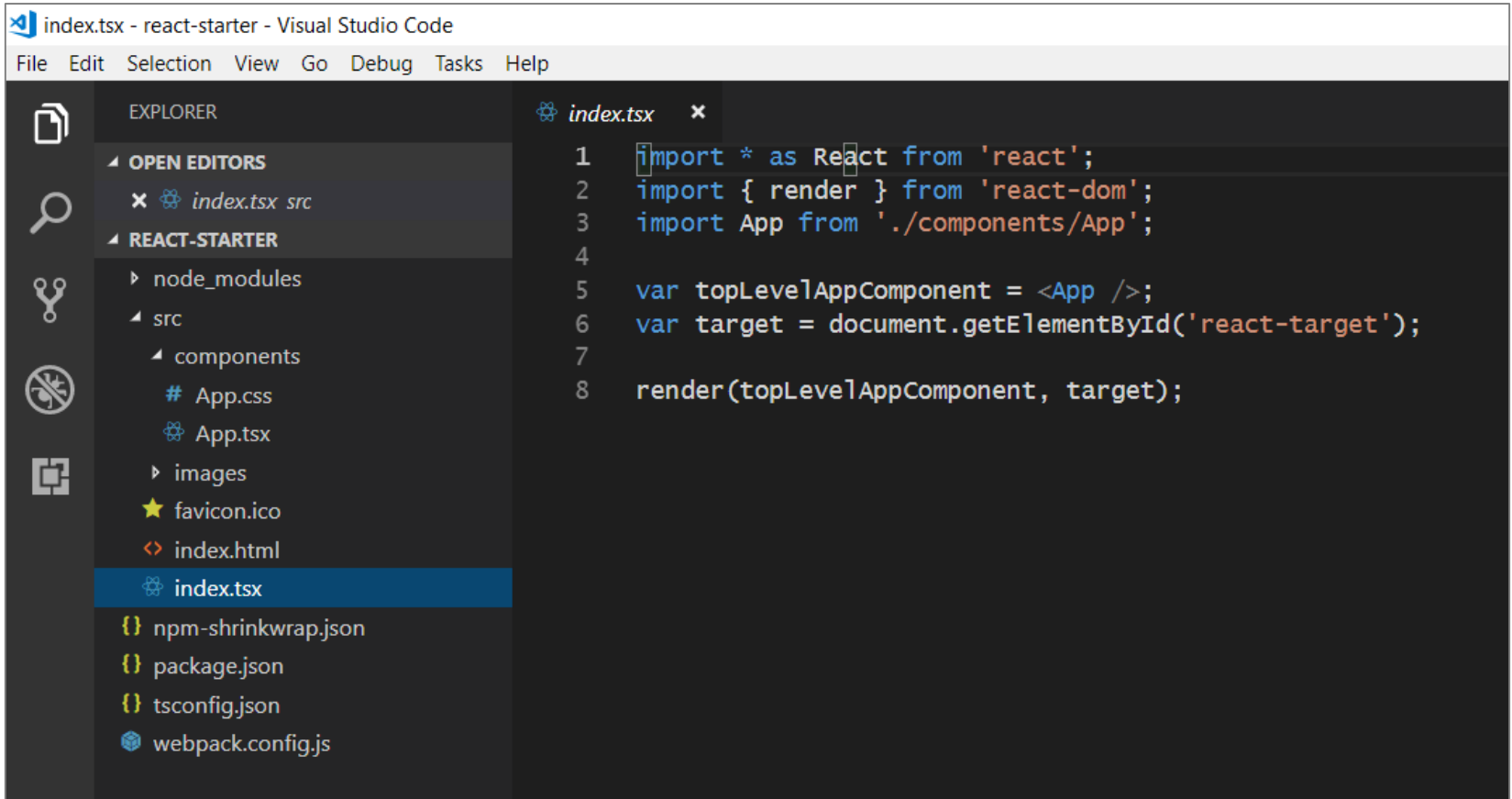
EXPLORER

- OPEN EDITORS
 - App.tsx src\components
- REACT-STARTER
 - node_modules
 - src
 - components
 - App.css
 - App.tsx**
 - images
 - favicon.ico
 - index.html
 - index.tsx
 - npm-shrinkwrap.json
 - package.json
 - tsconfig.json
 - webpack.config.js

```
1 import * as React from 'react';
2
3 import 'bootstrap/dist/css/bootstrap.min.css';
4 import 'bootstrap';
5
6 import './App.css';
7
8 export default class App extends React.Component<any, any> {
9
10   render() {
11
12     return (
13       <div id="page-container" className="container">
14         <div className="row navbar navbar-expand-sm navbar-dark bg-dark" role="navigation" >
15           <h1 style={{ 'color': 'white' }} >React Starter App</h1>
16         </div>
17         <div className="jumbotron">
18           <div>This is a sample starter app for with with React and TypeScript.</div>
19         </div>
20       </div>
21     );
22   }
23 }
24
25 }
```



Bootstrapping the App Component



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows the project structure for a React starter, with the `index.tsx` file selected. The code editor displays the following TypeScript code:

```
1 import * as React from 'react';
2 import { render } from 'react-dom';
3 import App from './components/App';
4
5 var topLevelAppComponent = <App />;
6 var target = document.getElementById('react-target');
7
8 render(topLevelAppComponent, target);
```



React Component Hierarchies

App.tsx - react-lab-exercise2 - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXPLORER

1

OPEN EDITORS 1 UNSAVED

- App.tsx src\components

REACT-LAB-EXERCISE2

- node_modules
- src
 - components
 - App.css
 - App.tsx
 - Banner.css
 - Banner.tsx
 - MainView.css
 - MainView.tsx
 - TopNav.css
 - TopNav.tsx
 - images
 - favicon.ico
 - index.html

App.tsx

```
import * as React from 'react';

import Banner from './Banner';
import TopNav from './Topnav';
import MainView from './MainView';

export default class App extends React.Component<any, any> {

  render() {
    return (
      <div id="page-container" className="container">
        <Banner appTitle="React Lab App" >
          <TopNav />
        </Banner>
        <MainView />
      </div>
    );
  }
}
```

App

Banner

Topnav

MainView

React Router

- Used to create route map in single page application (SPA)
 - Installed as a pair of npm packages
`npm install react-router @types/react-router --save-dev`
`npm install react-router-dom @types/react-router-dom --save-dev`
- Router must be added in as top-level component above App

```
index.tsx  x
import * as React from 'react';
import { render } from 'react-dom';
import App from './components/App';
import { HashRouter } from 'react-router-dom';

var topLevelAppComponent =
  <HashRouter>
    <App />
  </HashRouter>;

var target = document.getElementById('react-target');

render(topLevelAppComponent, target);
```




Using React Router

- Import Route and Switch components

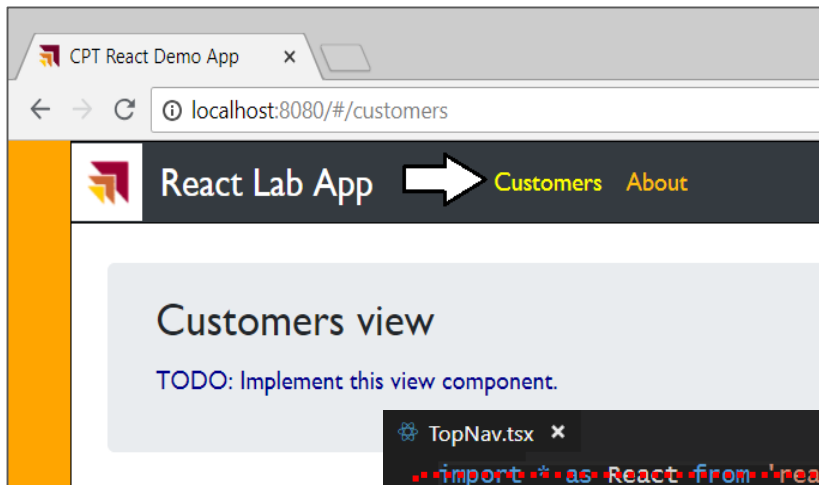
```
import * as React from 'react';  
import { Route, Switch } from 'react-router-dom';
```

- Create route map in HTML output

```
export default class App extends React.Component<any, any> {  
  
  render() {  
  
    return (  
      <div id="page-container" className="container">  
        <Banner appTitle="React Lab App" >  
          <TopNav />  
        </Banner>  
         <Switch>  
          <Route path="/" exact component={ViewHome} />  
          <Route path="/customers" component={ViewCustomers} />  
          <Route path="/about" component={ViewAbout} />  
        </Switch>  
      </div>  
    );  
  }  
}
```



Creating Route Links



```
TopNav.tsx x
import * as React from 'react';
import { Link, NavLink } from 'react-router-dom';
import './TopNav.css';

export default class TopNav extends React.Component<any, any> {

  render() {
    return (
      <div id="top-nav" className="navbar-collapse collapse" >
        <nav>
          <ul className="nav navbar-nav" >
            <li className="nav-item" >
              <NavLink exact to="/" className="navbar-link" activeClassName="active-nav-link" >
                Home
              </NavLink>
            </li>
            <li className="nav-item" >
              <NavLink to="/customers" className="navbar-link" activeClassName="active-nav-link" >
                Customers
              </NavLink>
            </li>
          </ul>
        </nav>
      </div>
    );
  }
}
```


Component Lifecycle

- `componentWillUpdate`
 - executed before component is rendered
- `componentDidUpdate`
 - executed after component is rendered
- `componentWillMount`
 - executed before node is added to the DOM
- `componentDidMount`
 - executed after node is added to the DOM
- `componentWillUnmount`
 - executed before node is removed from the DOM
- `shouldComponentUpdate(newProps, newState)`
 - executed before component is updated



Calling a Web Service using the Fetch API

```
getCustomers(): Promise<ICustomer[]> {  
  const restUrl =  
    "http://subliminalsystems.com/api/Customers/?" +  
    "$select=CustomerId,LastName,FirstName,EmailAddress,WorkPhone,HomePhone,Company" +  
    "&$filter=(CustomerId+1e+12)&$top=200";  
  return fetch(restUrl)  
    .then(response => response.json())  
    .then(response => {  
      console.log(response.value);  
      return response.value;  
    });  
}
```

```
getCustomer(customerId: string): Promise<ICustomerDetail> {  
  const restUrl = "http://subliminalsystems.com/api/Customers(" + customerId + ")";  
  return fetch(restUrl)  
    .then(response => response.json())  
    .then(response => {  
      console.log(response);  
      return response;  
    });  
}
```



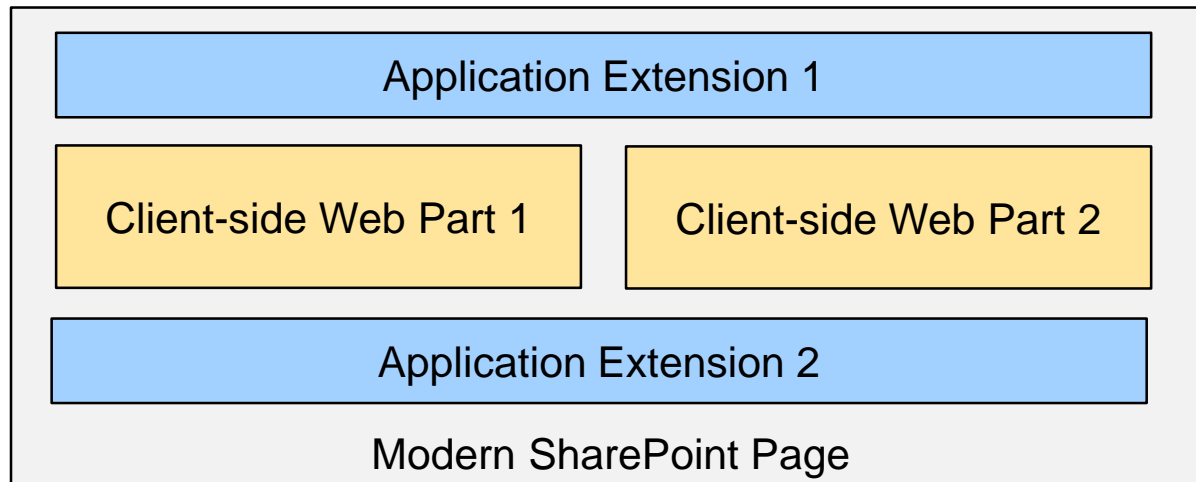
Agenda

- ✓ Introduction to Node.JS and NPM
- ✓ Automating Build Tasks using Gulp
- ✓ Bundling Project Assets using Webpack
- ✓ Developing with React.js, TypeScript and Webpack
- Developing with SharePoint Framework
- Developing Custom Visuals for Power BI



SharePoint Framework Component Types

- SPFx allows you to create several styles of webparts
 - Standard Webparts
 - React Webparts
- SPFx also provides several other Application Extensions
 - Application Customizer
 - Field Customizers
 - Command Sets



Installing Packages for SPFx Development

- Install Gulp (version 3)

```
npm install -g gulp
```

- Install Yeoman

```
npm install -g yo
```

- Install Yeoman Template for SPFx

```
npm install -g @microsoft/generator-sharepoint
```

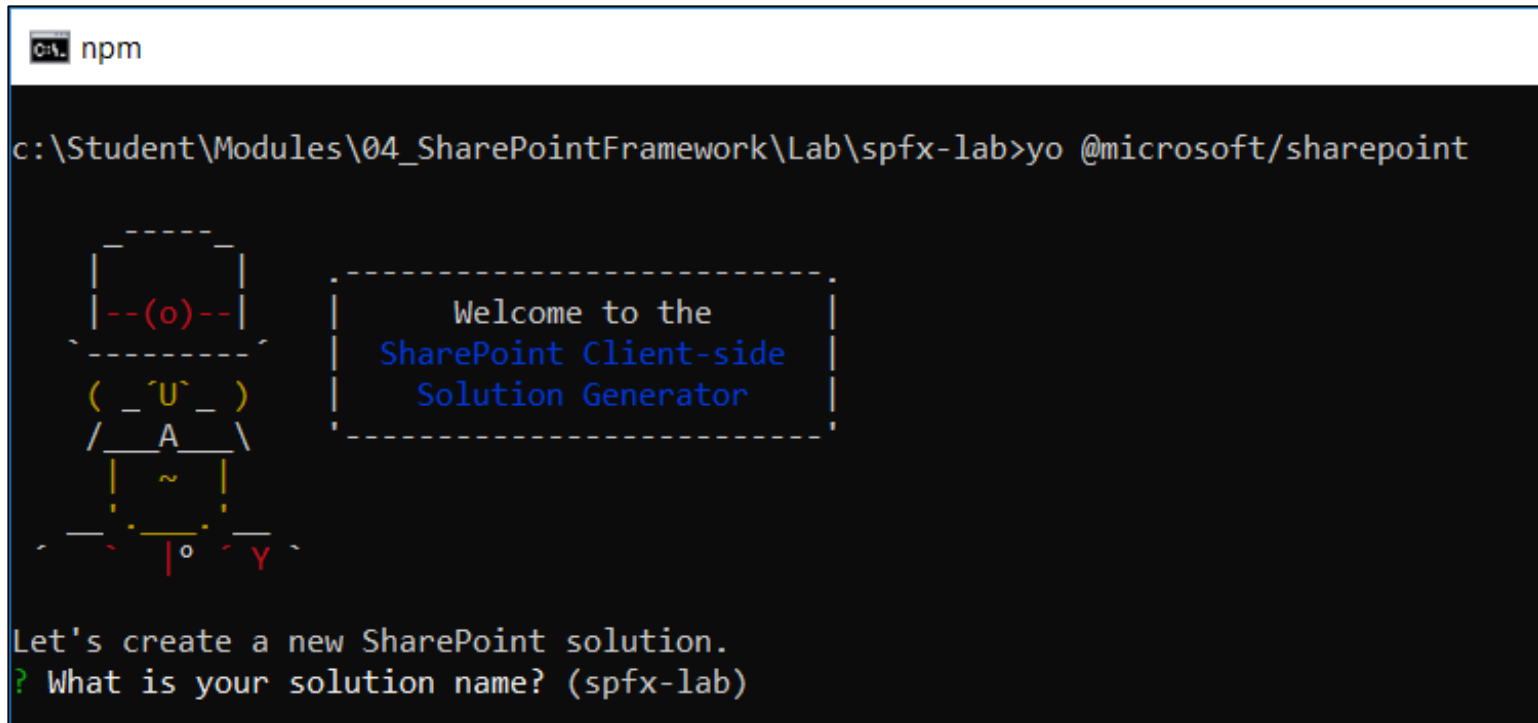


Using the SPFx Yeoman Template

- SPFx projects created with Yeoman template

```
yo @microsoft/sharepoint
```

- Template provides wizard-like experience when creating new project



The screenshot shows a terminal window with the title 'C:\ npm'. The command prompt shows the command `yo @microsoft/sharepoint` being executed in the directory `c:\Student\Modules\04_SharePointFramework\Lab\spfx-lab`. The output displays a ASCII art logo for the SharePoint Client-side Solution Generator, which includes a stylized 'U' and 'A' with a tilde '~' below them. To the right of the logo, a dashed box contains the text 'Welcome to the SharePoint Client-side Solution Generator'. Below the logo and text, the prompt 'Let's create a new SharePoint solution.' is shown, followed by the question '? What is your solution name? (spfx-lab)'.

```
C:\ npm
c:\Student\Modules\04_SharePointFramework\Lab\spfx-lab>yo @microsoft/sharepoint

  --(o)--
  |       |
  |  U   |
  |  A   |
  |  ~   |
  |  o   |
  |  Y   |

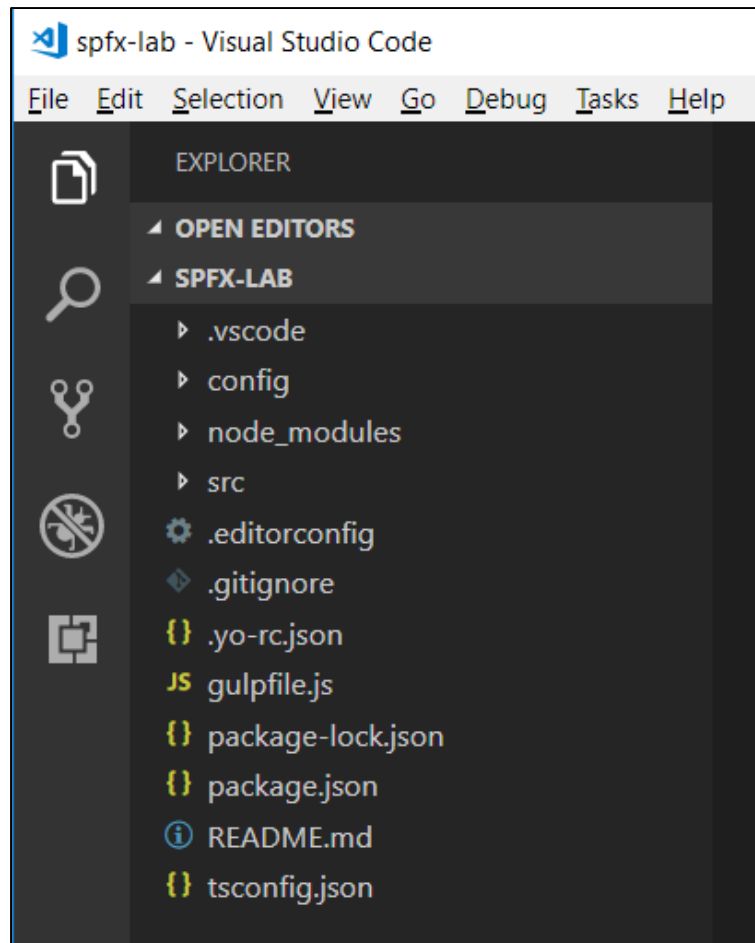
Welcome to the
SharePoint Client-side
Solution Generator

Let's create a new SharePoint solution.
? What is your solution name? (spfx-lab)
```



SharePoint Framework Project Structure

- Project created as Node.js project



SharePoint Framework Adds Gulp Tasks

- Run **gulp --tasks** to see SPFx gulp tasks added to project

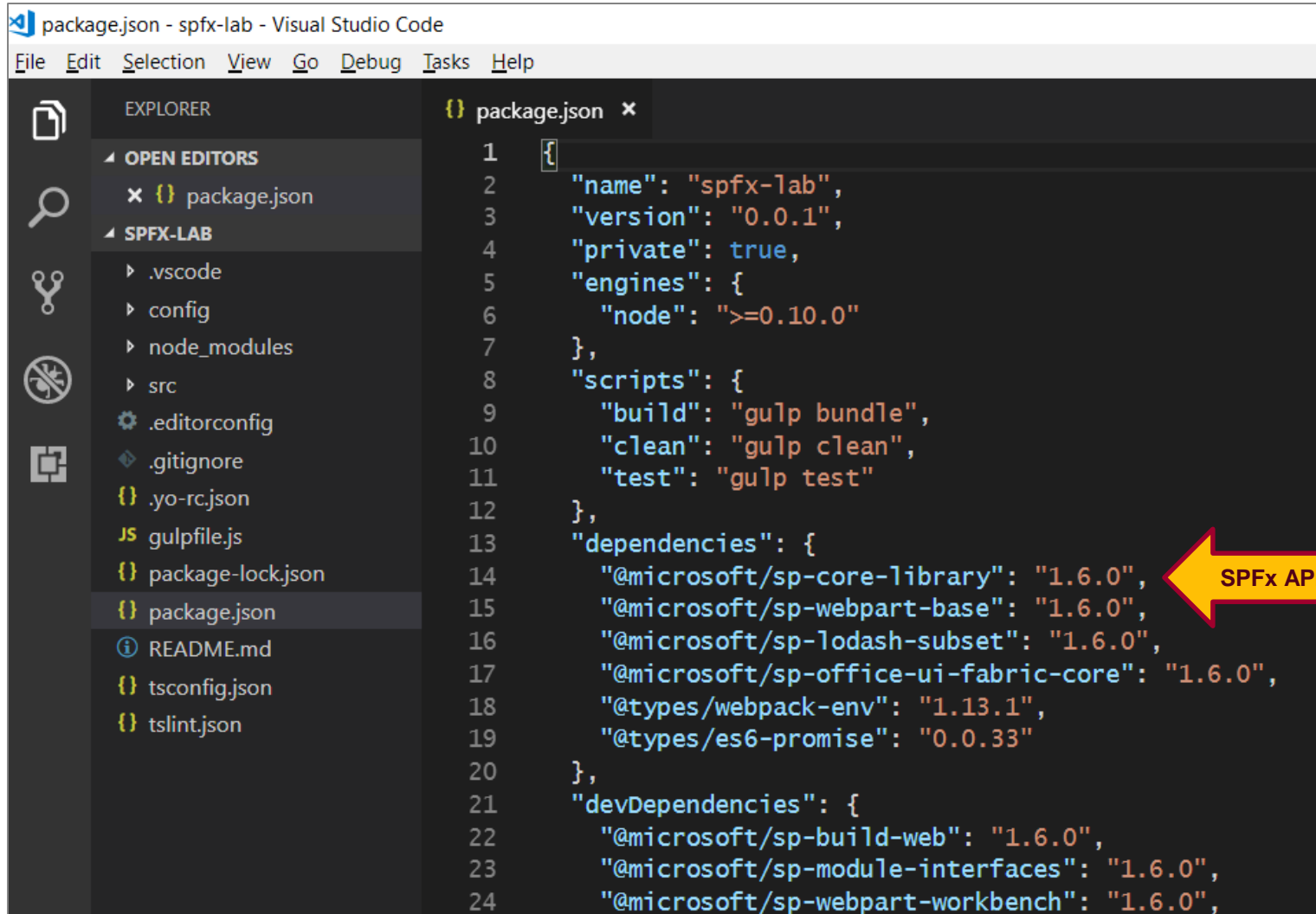
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab> gulp --tasks
[00:05:07] Using gulpfile C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab\gulpfile.js
[00:05:07] Tasks for C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab\gulpfile.js
[00:05:07] |— clean
[00:05:07] |— build
[00:05:07] |— default
[00:05:07] |— bundle
[00:05:07] |— dev-deploy
[00:05:07] |— deploy-azure-storage
[00:05:07] |— package-solution
[00:05:07] |— test
[00:05:07] |— serve
[00:05:07] |— trust-dev-cert
[00:05:07] |— untrust-dev-cert
PS C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab> █
```



Package.json



The screenshot shows the Visual Studio Code interface with the 'package.json' file open. The Explorer sidebar on the left lists the project files, including 'package.json'. The main editor area displays the JSON content of 'package.json' with line numbers 1 through 24. The JSON structure includes fields for 'name', 'version', 'private', 'engines', 'scripts', 'dependencies', and 'devDependencies'. A yellow arrow points from the text 'SPFx API Version Number' to the '1.6.0' version string in the '@microsoft/sp-core-library' dependency.

```
1  {
2    "name": "spfx-lab",
3    "version": "0.0.1",
4    "private": true,
5    "engines": {
6      "node": ">=0.10.0"
7    },
8    "scripts": {
9      "build": "gulp bundle",
10     "clean": "gulp clean",
11     "test": "gulp test"
12   },
13   "dependencies": {
14     "@microsoft/sp-core-library": "1.6.0",
15     "@microsoft/sp-webpart-base": "1.6.0",
16     "@microsoft/sp-lodash-subset": "1.6.0",
17     "@microsoft/sp-office-ui-fabric-core": "1.6.0",
18     "@types/webpack-env": "1.13.1",
19     "@types/es6-promise": "0.0.33"
20   },
21   "devDependencies": {
22     "@microsoft/sp-build-web": "1.6.0",
23     "@microsoft/sp-module-interfaces": "1.6.0",
24     "@microsoft/sp-webpart-workbench": "1.6.0",
```

SPFx API Version Number



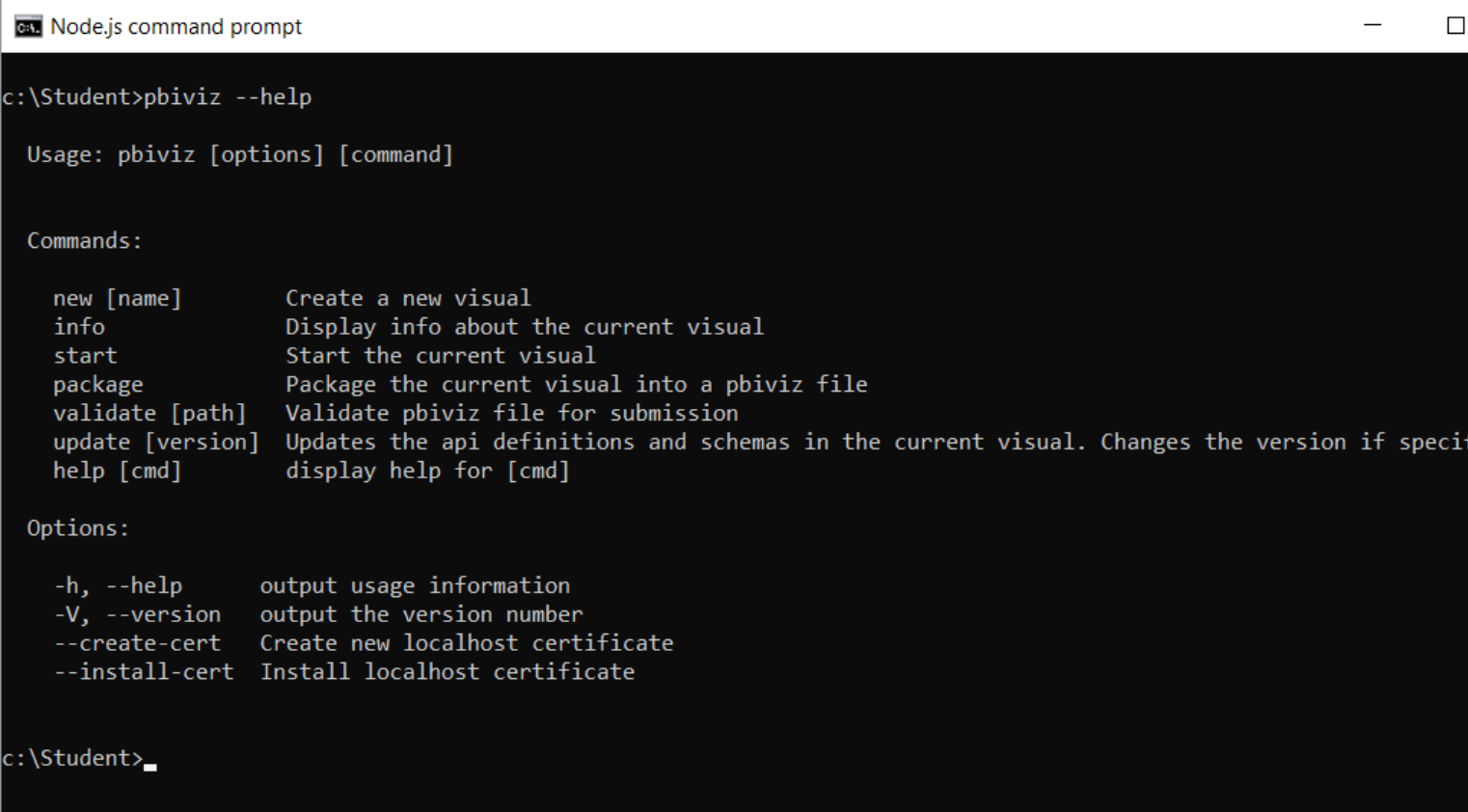
Agenda

- ✓ Introduction to Node.JS and NPM
- ✓ Automating Build Tasks using Gulp
- ✓ Bundling Project Assets using Webpack
- ✓ Developing with React.js, TypeScript and Webpack
- ✓ Developing with SharePoint Framework
- Developing Custom Visuals for Power BI



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>
```



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

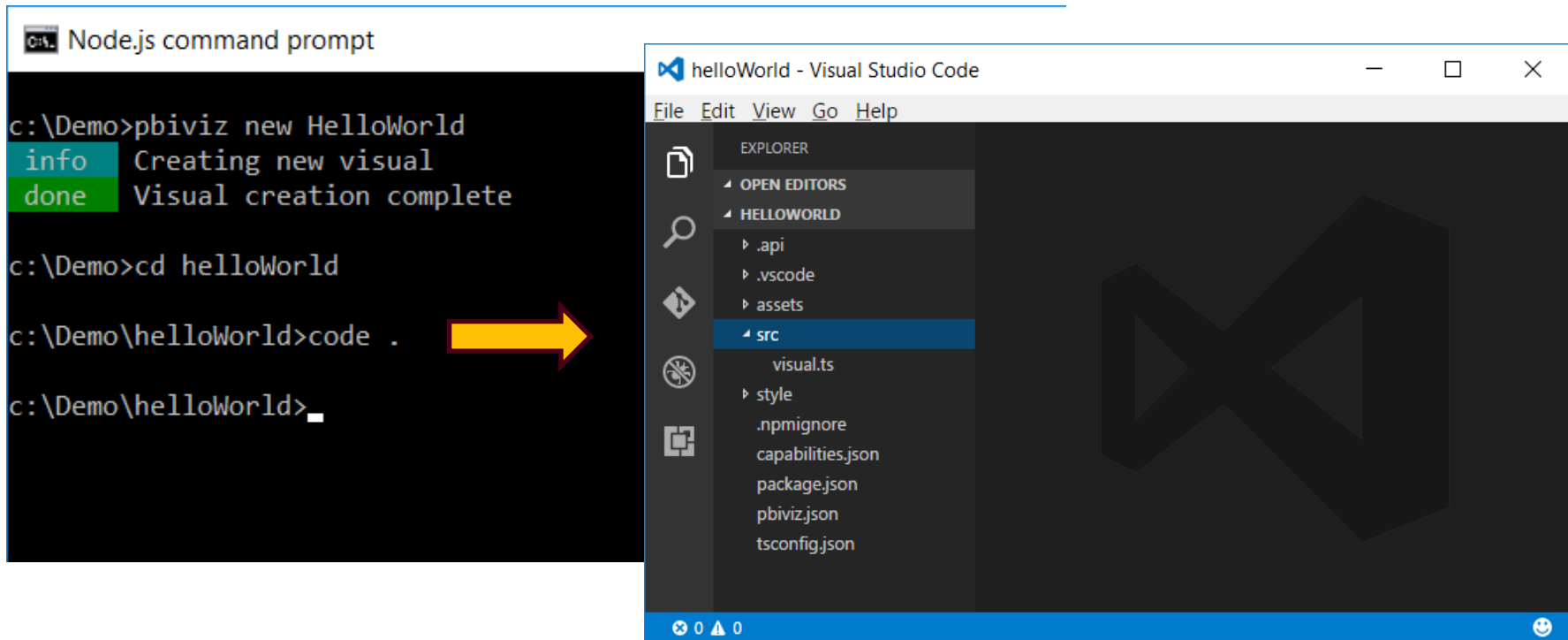
c:\Student>pbiviz --install-cert
info Use '15581865083792024' passphrase to install PFX certificate.

c:\Student>_
```



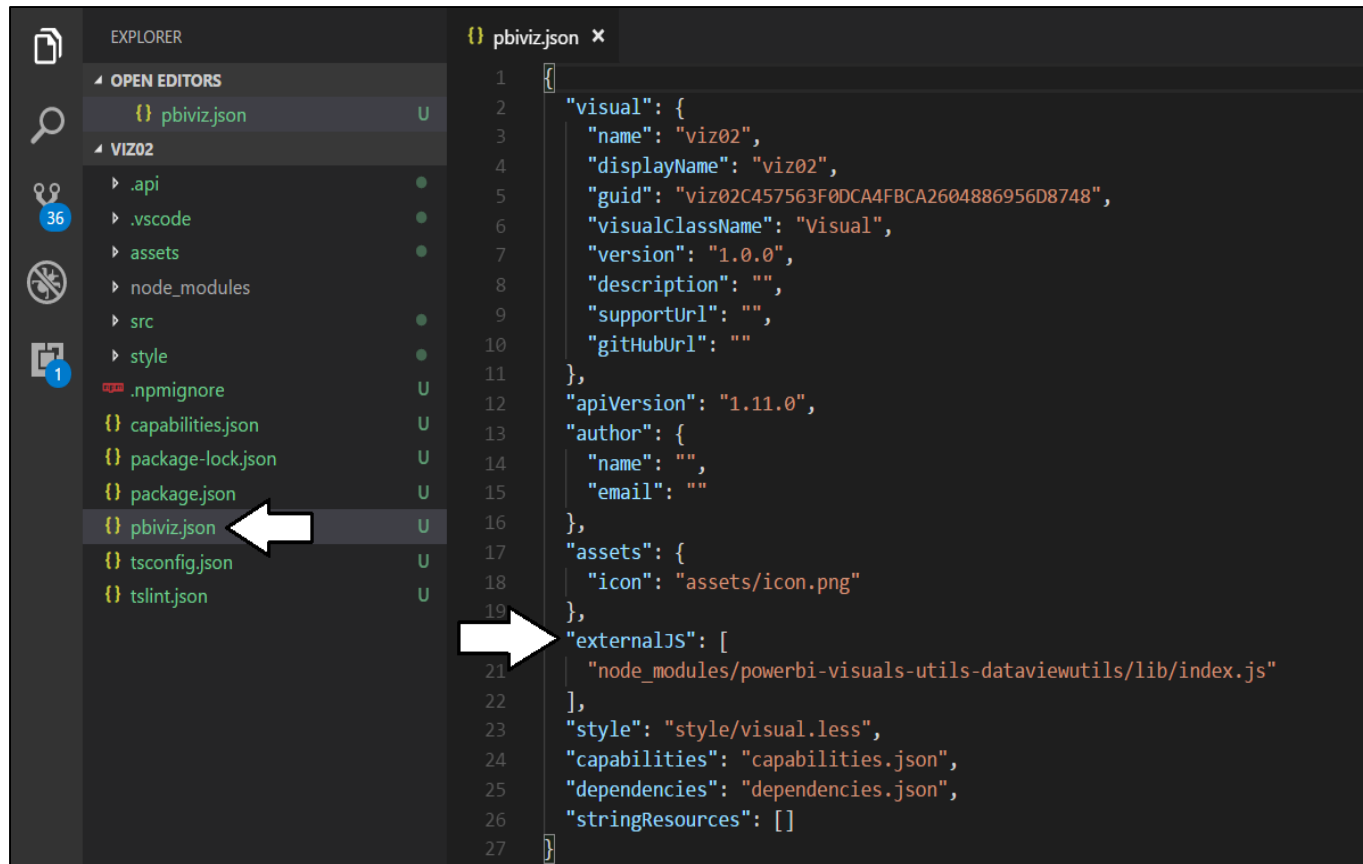
Creating a New Custom Visual Project

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



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 window on the right shows the content of 'pbiviz.json', with a white arrow pointing to the 'externalJS' array. The JSON content is as follows:

```
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 }
```



Summary

- ✓ Introduction to Node.JS and NPM
- ✓ Automating Build Tasks using Gulp
- ✓ Bundling Project Assets using Webpack
- ✓ Developing with React.js, TypeScript and Webpack
- ✓ Developing with SharePoint Framework
- ✓ Developing Custom Visuals for Power BI

