How have we been running JavaScript?

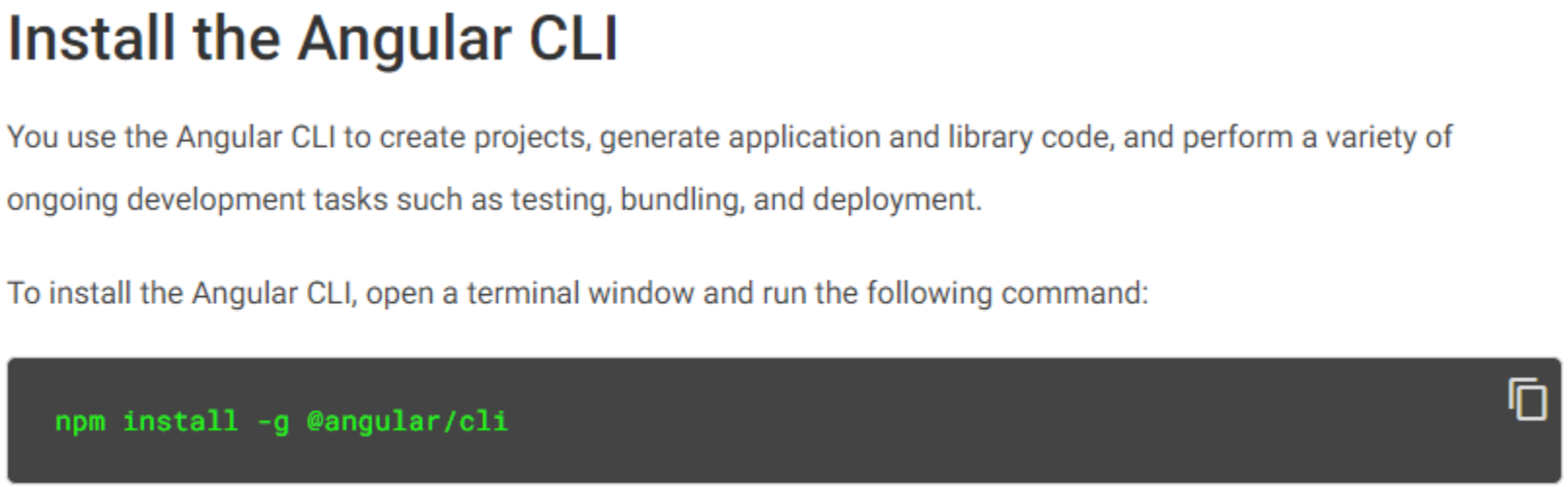
- Creating an html file and linking to a .js file
- This .js file is then executed
  - Browser is executing the JavaScript
  - If we're using Chrome, the V8 engine is what runs the JavaScript
- "**V8** is an open-source JavaScript engine developed by the Chromium Project for Google Chrome and Chromium web browsers"

Node.js

- Node.js is another runtime environment for JavaScript that exists outside of the browser
  - Utilizes the V8 engine as well
  - It's a way to execute JS without a browser
    - Ex: creating a backend server that is coded w/ JavaScript as the language
- It's also something that we commonly utilize with other tools in order to build various JavaScript based projects (such as Angular projects)
  - We will heavily utilize a tool called npm (node package manager)

Angular CLI

- Angular CLI = Angular Command line interface
  - When we install Angular CLI (using npm, node package manager), gives us the **ng** command that allows us to quickly create Angular projects and different “pieces” that we need for an Angular application
  - The –g flag when running the `npm install –g @angular/cli` basically is installing this CLI to our entire machine. –g = global



Creating an Angular Project

- ng new <project name>
- This will generate all of the files that form the “skeleton” of our Angular project

Important files/directories inside of Project folder

- node_modules (folder/directory): contains all of the dependencies that the Angular project requires
  - Any dependencies that you install using `npm install` will be placed into the node_modules folder
  - node_modules can be large in size
    - So, we should not be pushing the node_modules folder to github
- package.json: contains a list of all the dependencies that need to be installed (npm related file)
  - Analogous to the pom.xml file
  - When we are inside of the directory that contains the package.json file, we can run `npm install` with no other arguments to go ahead and install all of the necessary dependencies
  - Two types of dependencies:
    - dependencies: dependencies required in the final application
    - devDependencies: dependencies that are only required when you are developing the application (testing dependencies, etc.)
  - There is a scripts section inside of the file that will contain different “scripts” that you can run using `npm run <script name>`
    - These scripts will correspond to particular commands such as the **start** script being associated with **ng serve**
    - **Ex.** npm run start

Important files/directories inside of Project folder (continued)

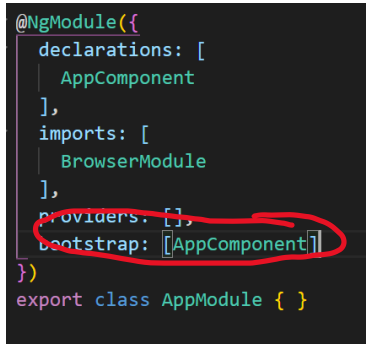
- package-lock.json: stores an “exact, versioned dependency tree”
  - Detailed version of package.json
- angular.json: contains different settings for the angular project
- src directory: contains the actual code that we will be working with primarily when developing an Angular application
  - HTML, TS, CSS
  - Index.html: this is the html file that will form the actual “single page” for our SPA (single page application)
  - main.ts: code that initially executes when first starting up an Angular application
  - styles.css: global styling file
  - assets directory: images and whatever other files you want to include will be stored here
  - App directory: the folder that contains our initial main **component** in the application

Angular Bootstrapping Process

- When the index.html first loads up, the script tags for the .js files will be executed
  - The .js files contain all of the HTML, CSS, and other logic that is defined in the different components that we have
- The code inside of main.ts is then executed. In particular, a module known as the AppModule is loaded up “or bootstrapped”

```
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

- The AppModule file contains a property called bootstrap, which contains the AppComponent that needs to be instantiated and loaded into the DOM



Angular Boostrapping Process (Continued)

- Once the AppComponent is loaded up based on the boostrap property inside of AppModule, the “selector” property inside of AppComponent itself will be identified

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-angular-project';
}
```

- Going back to the index.html, there is a tag called <app-root></app-root>
  - This directly corresponds with the “selector” property defined in the AppComponent
  - Therefore, Angular will take this component and render it in-place of the app-root tag

```
  <meta name="viewport" con
  <link rel="icon" type="im
</head>
<body>
  <app-root></app-root> <!-
  <!-- This tag represents
  for our Angular applicati
</body>
</html>
```

Angular Component files

- 4 different files associated with an Angular component
  1. <component name>.component.html
     - Contains the HTML for our particular component
  2. <component name>.component.css
     - Contains the CSS for our particular component
  3. <component name>.component.ts
     - Contains all of the programming logic for our component
     - Utilizing TypeScript (JavaScript but with strong typing)
  4. <component name>.component.spec.ts
     - For writing unit tests on our methods in our component.ts file