



# **3416-003 Web Development Using Angular 8 and TypeScript**

## **Module 1: Introduction to AngularJS**



## Module 1: Section 1

# What Is This Course About



# Learning Outcomes for this Module

- What is Angular?
- Angular vs Angular 2 vs Angular 7
- CLI Deep Dive & Troubleshooting
- Project Setup and First App
- Editing the First App
- What is TypeScript?
- A Basic Project Setup using Bootstrap for Styling
- Installation of NodeJS
- Installation of CLI
- Installation of TypeScript
- Installation of Microsoft Visual Studio Code

# What is Angular?

- Angular is a JavaScript Framework which allows you to create reactive Single-Page-Applications(SPAs).
- What exactly does it mean?
- A Single Page Application (SPA) is an application like the one shown here for your group project. You can navigate around and in the URL, you can see that we seem to visit different pages, but in the end, our page never changes.
- It's only one HTML file and a bunch of JavaScript code we got from the server and everything which you see, every change, is rendered in the browser.

# What is Angular?

- Now, why is that awesome?
- It gives the user a very reactive user experience(UX).
- JavaScript is much faster than having to reach out to a server for every page change and for every new piece of data you want to display. Therefore, this approach allows you to create web applications which look and feel almost like mobile applications; very fast!
- Everything happens instantly. If you do need some data from a server, you simply load it in the background so that the user never leaves this experience of having a reactive web application to use.
- So every click I do here for the group project simply changes this one single page we're using; this one HTML page.
- So, how is this done?

# What is Angular?

- Well, JavaScript changes the DOM, changes whatever is displayed here(in the browser), by changing the HTML code during runtime(so to say). That is why you never see the refresh icon the top-left spin; because we're only changing the currently-loaded page. You can even see that if you inspect the source code of a page like this.
- That is the html file and as you can see, it doesn't seem to contain the content you are seeing on this page.

# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- Maybe you also came across the many words of angular that you can find there is angular 8 now.
- We also had six.
- Couple of words to an angular one.
- Now angular one was initially released in 2010 and it provided a revolution to the way we can build front end user interfaces in the browser because it was a huge step forward when it came to using browser side javascript to re-rendered the DOM, update the dom at runtime and therefore provide highly interactive user experiences without reloading the page.
- Angular 2 was a complete rewrite of Angular 1. Angular 2 was initially released in 2016.

# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- It has nothing in common with angular one except for the core team that developed it.
- And therefore if you know angular one unfortunately that knowledge will not help you much.
- On the other hand, you also don't need to know angular one to get the most out of this course.
- Now Angular 2, Angular 4, Angular 5, Angular 6, Angular 7 and Angular 8 is just referred as Angular. It's one in the same framework which totally differs from Angular One but which doesn't differ a lot within Angular 2, Angular 5, Angular 5, Angular 6, Angular 7 and Angular 8. So this is just Angular, on the other hand Angular One is now called AngularJS to make it clear that these are two totally separate frameworks now two different words.

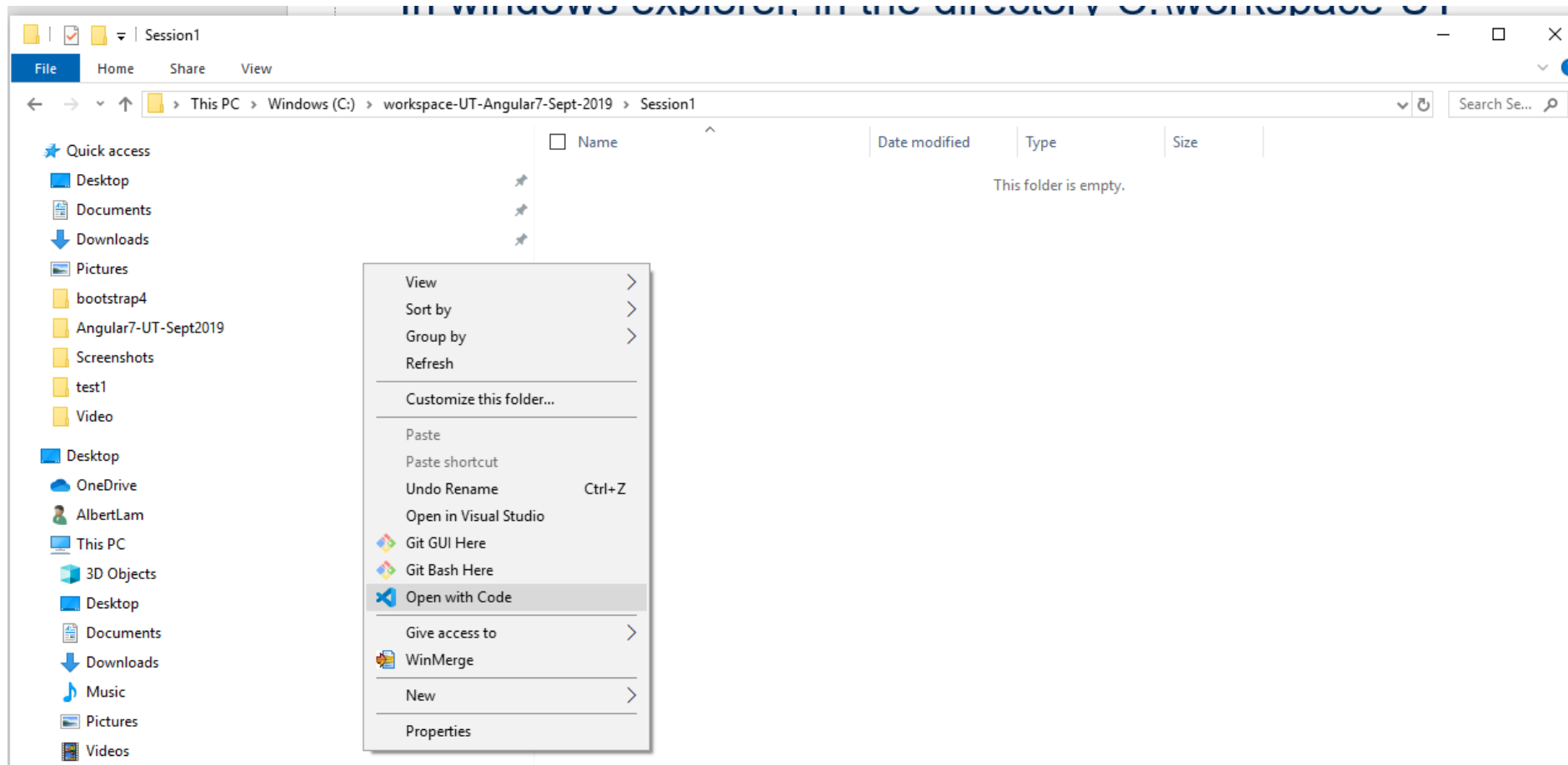


# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- So the different versions of Angular 2 plus are really just incremental improvements.
- No complete rewrites.
- The syntax we used in Angular 8 and therefore in this course is the same as we used in Angular 2.
- There were just some bug fixes, minor improvements, some new features but generally the syntax is the same.
- Go to Angular8-UT-SEPT-2020\SoftwareToDownload\Software-To-Download-Version3.pDF
- Create directory on c drive:C:\ workspace-UT-Angular8-SEPT-2020\Session1

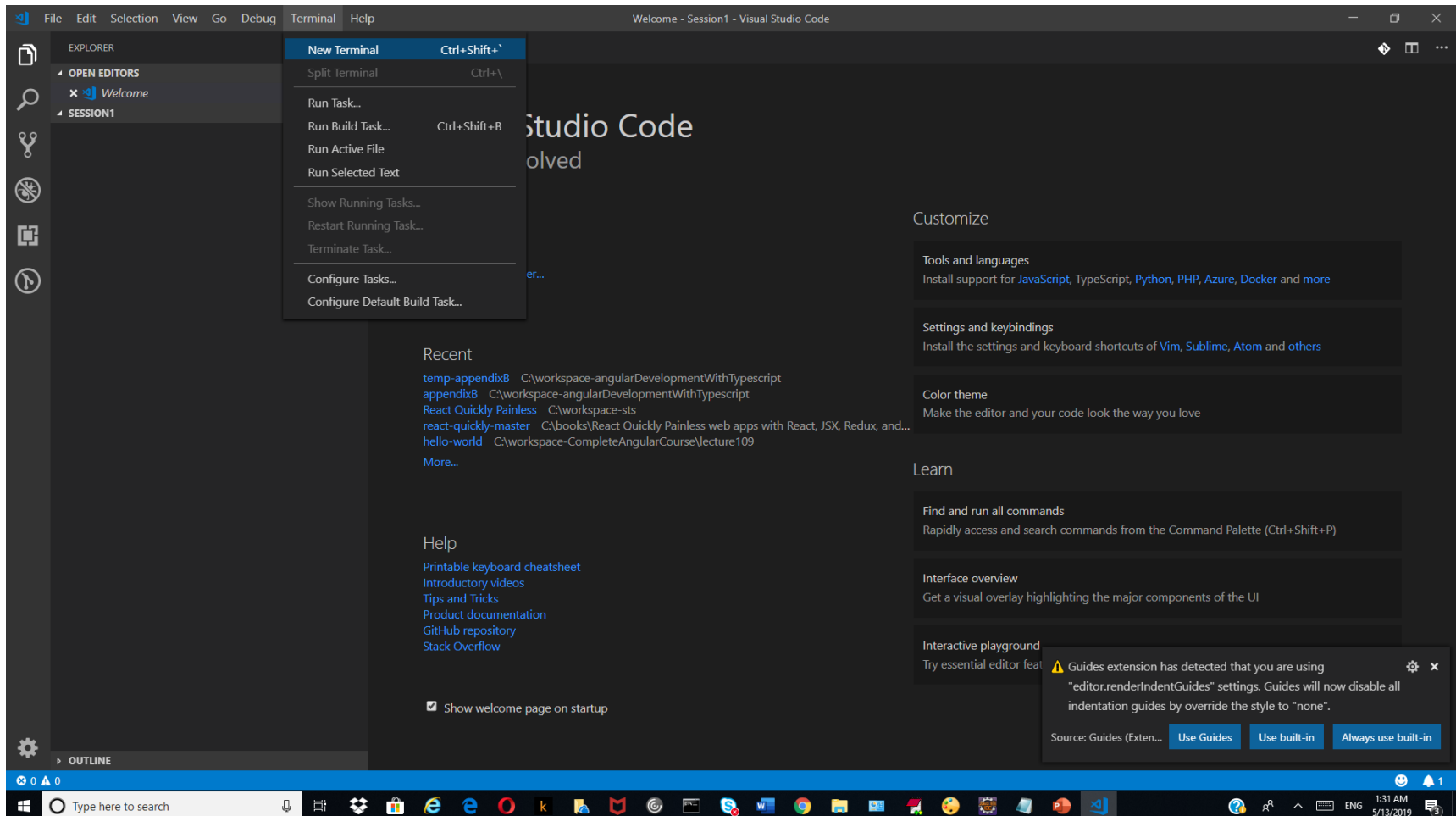
# My First App

- In windows explorer, in the directory C:\workspace-UT-Angular8-SEPT-2020\Session1, right click and choose “Open with code”.



# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- In Visual Studio Code, choose “Terminal/New Terminal”:



# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- In Terminal, run the following command:

```
C:\workspace-UT-Angular8-SEPT-2020\Session1> ng new my-dream-app
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

1: node ▼



```
PS C:\workspace-UT-Angular7-Sept-2019\Session1> ng new my-dream-app
```

```
? Would you like to add Angular routing? No
```

```
? Which stylesheet format would you like to use? (Use arrow keys)
```

```
> CSS
```

```
SCSS [ http://sass-lang.com/documentation/file.SASS_REFERENCE.html#syntax ]
```

```
? Which stylesheet format would you like to use? CSS
```

# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- In Terminal, run the following command:

```
C:\workspace-UT-Angular8-SEPT-2020\Session1> ng new  
my-dream-app
```

```
>Would you like to add Angular routing?No
```

```
Which stylesheet format would like to use?click enter for  
default.
```

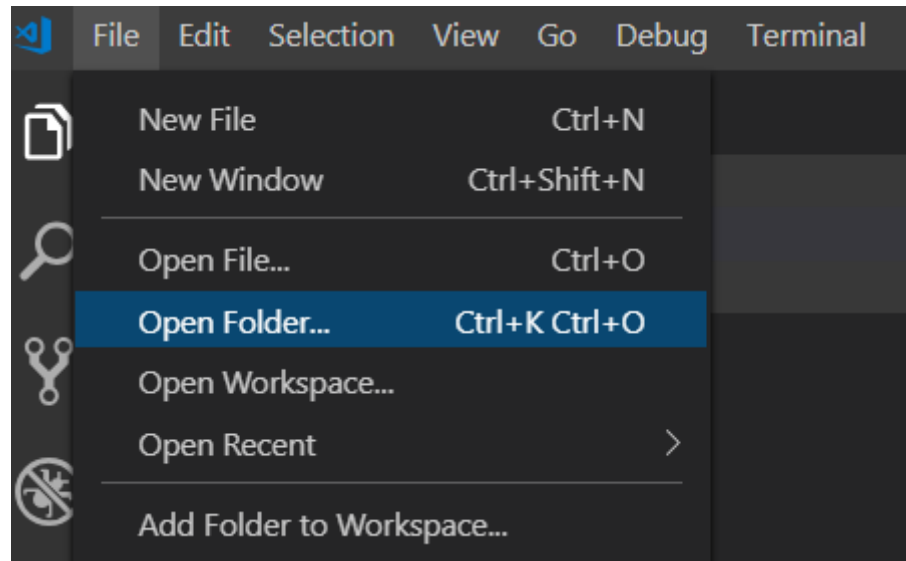
# DOS-PROMPT

- If Microsoft Visual Studio Code Terminal doesn't work. Go to dos-prompt: C:\workspace-UT-Angular8-SEPT-2020\Session1
- C:\workspace-UT-Angular8-SEPT-2020\Session1>ng new my-dream-app

```
C:\workspace-UT-Angular8-JAN-2020\Session1>ng new my-dream-app
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE my-dream-app/angular.json (3641 bytes)
```

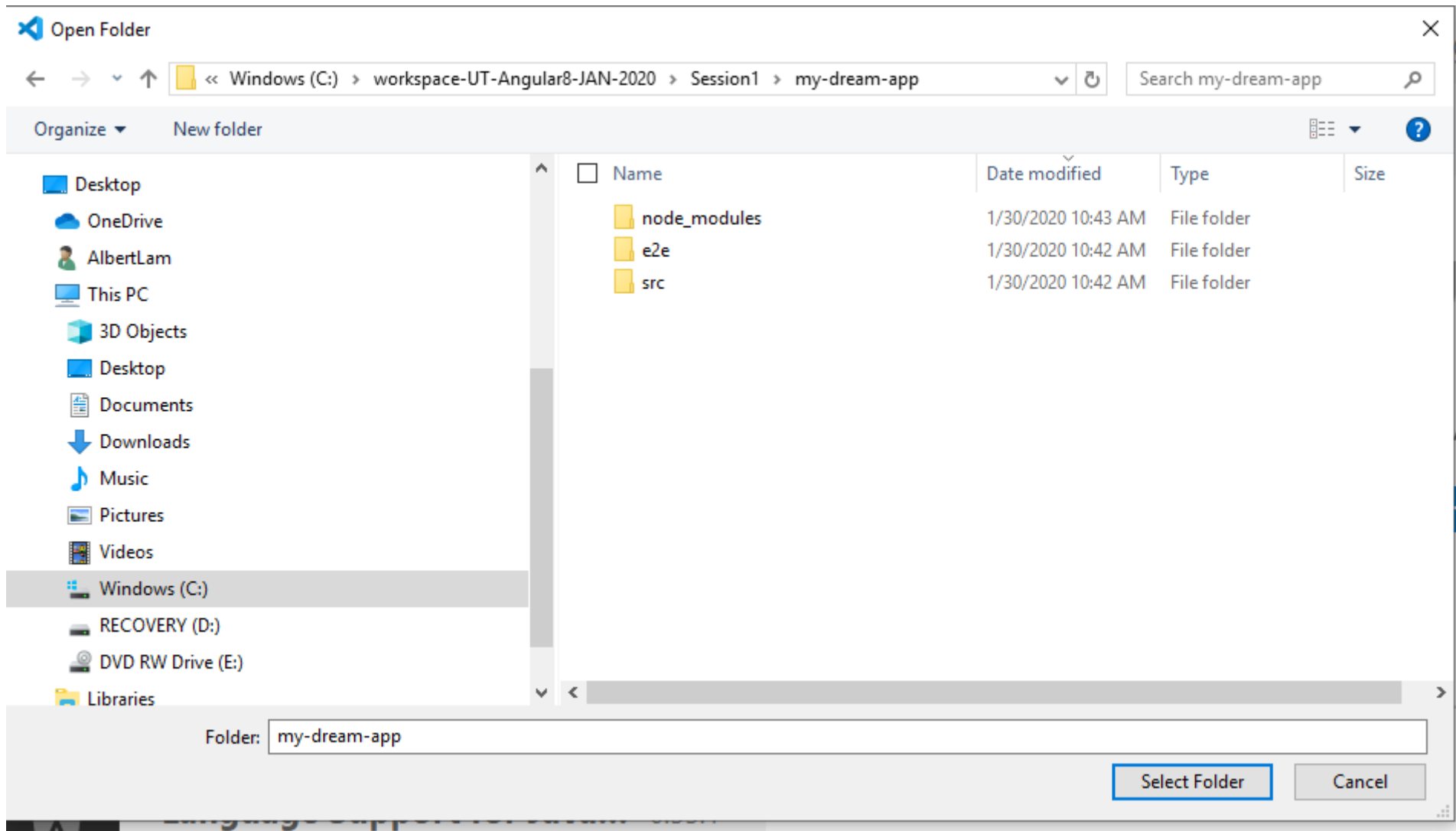
# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- Click “File/Open Folder”:



# Angular 8 vs Angular 7 vs Angular 6 vs Angular 2 vs Angular 1

- Choose “My Dream App” and click “Select Folder”.





# My First App

- \$ cd my-dream-app
- \$ng serve

```
C:\workspace-UT-Angular8-JAN-2020\Session1>cd my-dream-app  
C:\workspace-UT-Angular8-JAN-2020\Session1\my-dream-app>ng serve -o
```

# My First App

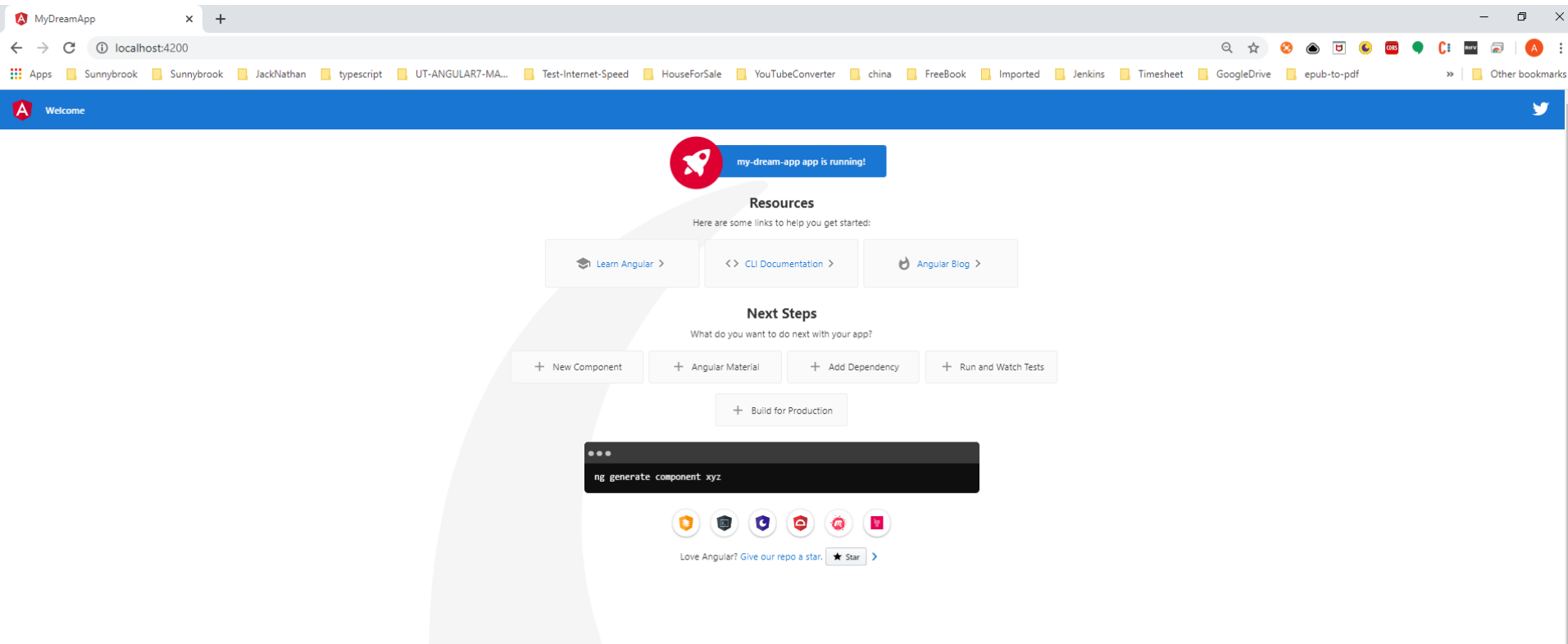
- \$ cd my-dream-app
- \$ng serve

```
C:\workspace-UT-Angular8-JAN-2020\Session1>cd my-dream-app
C:\workspace-UT-Angular8-JAN-2020\Session1\my-dream-app>ng serve -o
10% building 3/3 modules 0 active i @wds@: Project is running at http://localhost:4200/webpack-dev-server/
i @wds@: webpack output is served from /
i @wds@: 404s will fallback to //index.html

chunk {main} main.js, main.js.map (main) 47.9 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 264 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.73 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.81 MB [initial] [rendered]
Date: 2020-01-30T15:47:52.580Z - Hash: 5ae19ad9b959ed150a14 - Time: 8672ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
i @wdm@: Compiled successfully.
```

# My First App

- <http://localhost:4200/>



# Editing the First App

- Click package.json.
- Now this is the project loaded into the IDE.
- These are all the folders and files the Angular CLI created for you. This is your entire Angular project.
- Now this might look intimidating because you've got so many files in there. Most of these files are just doing some configuration work and you don't really need to touch them.
- One interesting file is the package.json file.

# Editing the First App

- Here you can see all the dependencies of your project like Angular 8 and these are third-party packages your project needs to run correctly.

```
"dependencies": {  
  "@angular/animations": "~8.2.14",  
  "@angular/common": "~8.2.14",  
  "@angular/compiler": "~8.2.14",  
  "@angular/core": "~8.2.14",  
  "@angular/forms": "~8.2.14",  
  "@angular/platform-browser": "~8.2.14",  
  "@angular/platform-browser-dynamic": "~8.2.14",  
  "@angular/router": "~8.2.14",  
  "rxjs": "~6.4.0",  
  "tslib": "^1.10.0",  
  "zone.js": "~0.9.1"
```

# Editing the First App

- All devDependencies are only required for development.

```
"devDependencies": {  
  "@angular-devkit/build-angular": "~0.803.19",  
  "@angular/cli": "~8.3.19",  
  "@angular/compiler-cli": "~8.2.14",  
  "@angular/language-service": "~8.2.14",  
  "@types/node": "~8.9.4",  
  "@types/jasmine": "~3.3.8",  
  "@types/jasminewd2": "~2.0.3",  
  "codelyzer": "^5.0.0",  
  "jasmine-core": "~3.4.0",  
  "jasmine-spec-reporter": "~4.2.1",  
  "karma": "~4.1.0",  
  "karma-chrome-launcher": "~2.2.0",  
  "karma-coverage-istanbul-reporter": "~2.0.1",  
  "karma-jasmine": "~2.0.1",  
  "karma-jasmine-html-reporter": "^1.4.0",  
  "protractor": "~5.4.0",  
  "ts-node": "~7.0.0",  
  "tslint": "~5.15.0",  
  "typescript": "~3.5.3"  
}
```

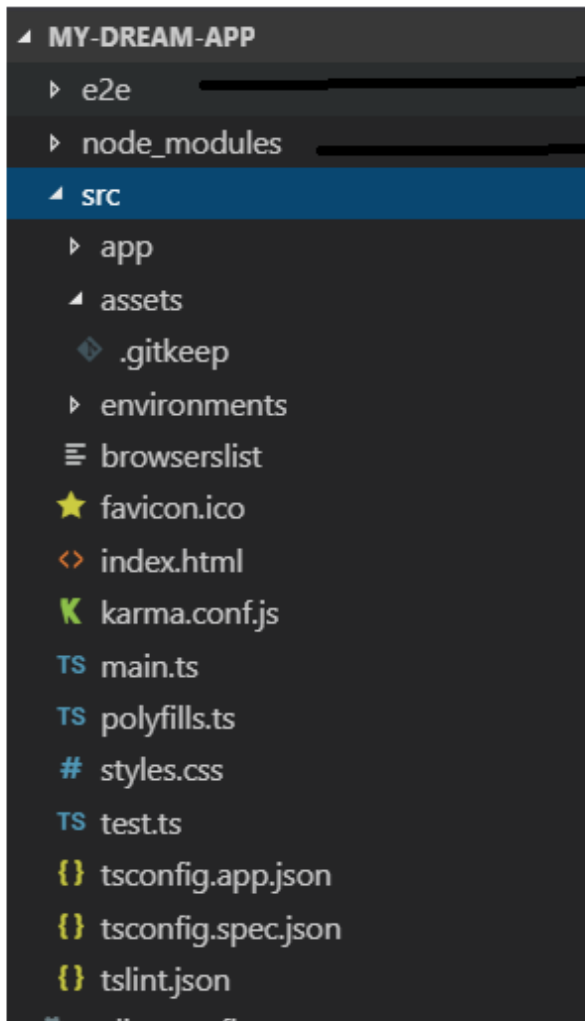
# Editing the First App

- Click package.json:

```
13  ✓  "dependencies": {
14      "@angular/animations": "~8.2.14",
15      "@angular/common": "~8.2.14",
16      "@angular/compiler": "~8.2.14",
17      "@angular/core": "~8.2.14",
18      "@angular/forms": "~8.2.14",
19      "@angular/platform-browser": "~8.2.14",
20      "@angular/platform-browser-dynamic": "~8.2.14",
21      "@angular/router": "~8.2.14",
22      "rxjs": "~6.4.0",
23      "tslib": "^1.10.0",
24      "zone.js": "~0.9.1"
25  },
26  ✓  "devDependencies": {
27      "@angular-devkit/build-angular": "~0.803.19",
28      "@angular/cli": "~8.3.19",
29      "@angular/compiler-cli": "~8.2.14",
30      "@angular/language-service": "~8.2.14",
31      "@types/node": "~8.9.4",
32      "@types/jasmine": "~3.3.8",
33      "@types/jasminewd2": "~2.0.3",
34      "codelyzer": "^5.0.0",
35      "jasmine-core": "~3.4.0",
36      "jasmine-spec-reporter": "~4.2.1",
37      "karma": "~4.1.0",
38      "karma-chrome-launcher": "~2.2.0",
39      "karma-coverage-istanbul-reporter": "~2.0.1",
40      "karma-jasmine": "~2.0.1",
41      "karma-jasmine-html-reporter": "^1.4.0",
42      "protractor": "~5.4.0",
43      "ts-node": "~7.0.0",
44      "tslint": "~5.15.0",
```

# Editing the First App

- So let's jump into the src folder because that is where our code is.



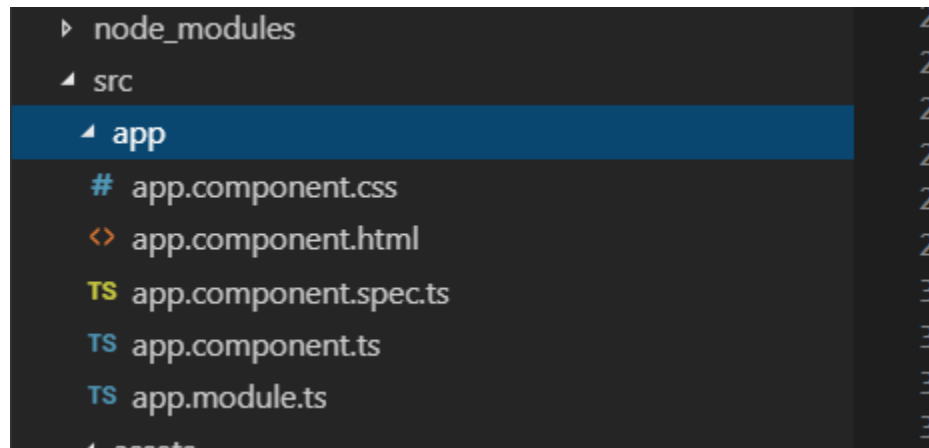
e2e is for end-to-end testing(we'll ignore this)

node\_modules is where all these dependencies you see in the package.json file actually were installed.



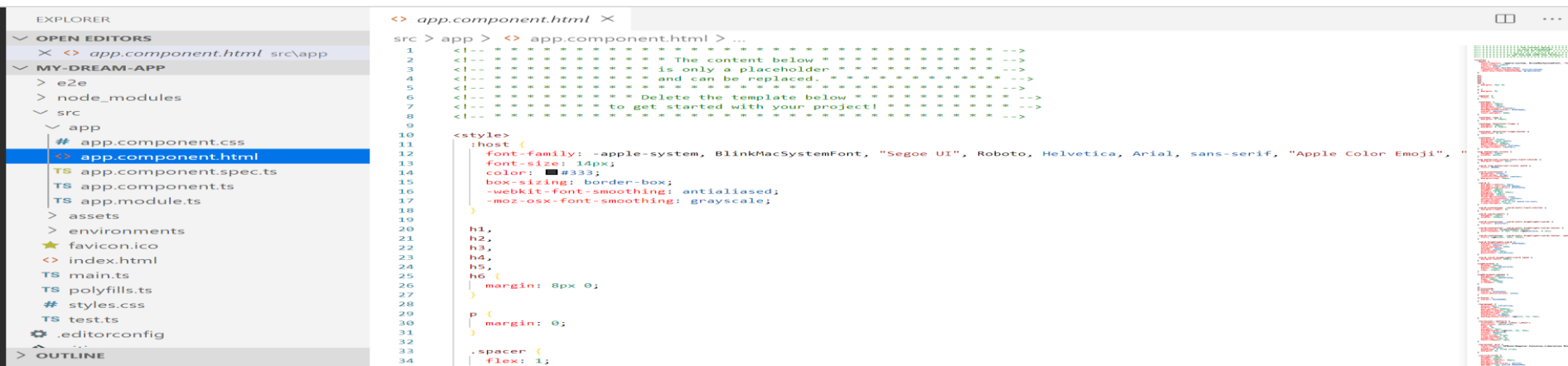
# Editing the First App

- So let's jump into the src folder because that is where our code is.
- We've got this app folder, under src, and in this app folder we see some other files.



# Editing the First App

- For now let's open the app.component.html file.
- We've got this app folder and in this app folder we see some other files.
- For now let's open the app.component.html file.
- Here we indeed see something which looks deceptively like what we saw in the browser.



# Editing the First App

MyDreamApp x +

localhost:4200

Apps Sunnybrook Sunnybrook JackNathan typescript UT-ANGULAR7-MA... Test-Internet-Speed HouseForSale YouTubeConverter china FreeBook Imported Jenkins Timesheet Google

Welcome

my-dream-app app is running!

### Resources

Here are some links to help you get started:

- [Learn Angular >](#)
- [<> CLI Documentation >](#)
- [Angular Blog >](#)

### Next Steps

What do you want to do next with your app?

- [+ New Component](#)
- [+ Angular Material](#)
- [+ Add Dependency](#)
- [+ Run and Watch Tests](#)
- [+ Build for Production](#)

```
ng generate component xyz
```

Love Angular? [Give our repo a star.](#) [★ Star >](#)

# Editing the First App

- Change app.component.html

- Original:

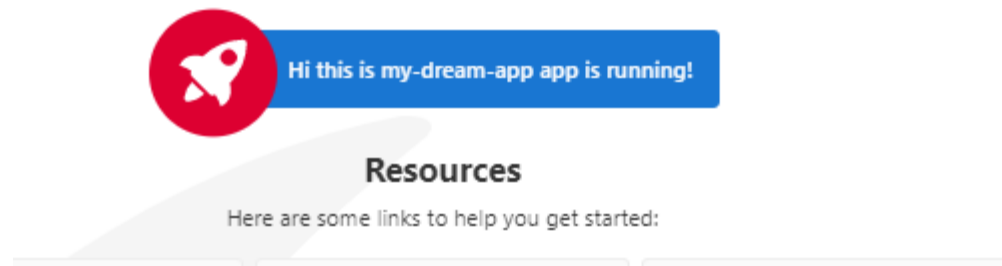
```
<span>{{ title }} app is running!</span>
```

- New:

```
<span>Hi this is {{ title }} app is running!</span>
```

# Editing the First App

- Click “ctrl s” to save.
- See title = “my-dream-app”.



# Editing the First App

- Click “ctrl s” to save.
- Now one strange thing we see is this “my-dream-app” here, but we actually only see these curly braces {{ title}} and title in the app component html file and therefore, we can already see some of the work Angular does here.
- Angular is, of course, not a tool to allow us to write static HTML files. We wouldn't need a framework for that. It allows us to mix static HTML code and dynamic things we want to output in that code and actually, what we have here is one of these components Angular works with; the app component. A component always has a template, the HTML code, possibly has some styling in the CSS file(though it's empty here as you can see).

**Hi, this: my-dream-app!**

# Editing the First App

- The file `app.component.ts` is the typescript file for this module. If we enter this, this is Typescript and this is now the definition of the component. This is what will be converted to normal JavaScript by the build workflow.
- And in this file, we see a couple of interesting things like `@Component`, I'll come back to that and what in detail is happening here in the next session.

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'my-dream-app';
10 }
11
```

# Editing the First App

- We also see title = 'my-dream-app'. Now what's that? If you go back to the app.component.html file, we also saw title here right; between the curly braces. So an assumption would be that the title in the Typescript file is related to that.
- So let's maybe change this to "my app" and now if we save this it recompiles and if we go back, we see "Hi, this is: my app!". Now this might not be perfect English, but we can definitely see that something changed.

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'my-dream-app';
10 }
11
```

```
<h1>
  Hi, this: {{ title }}!
</h1>
```



# Editing the First App

- This is so-called data binding in action; a concept we'll also dive into deeply in this course. This is how we can output dynamic content. This could of course be content that is calculated dynamically or retrieved from a server in our HTML code.

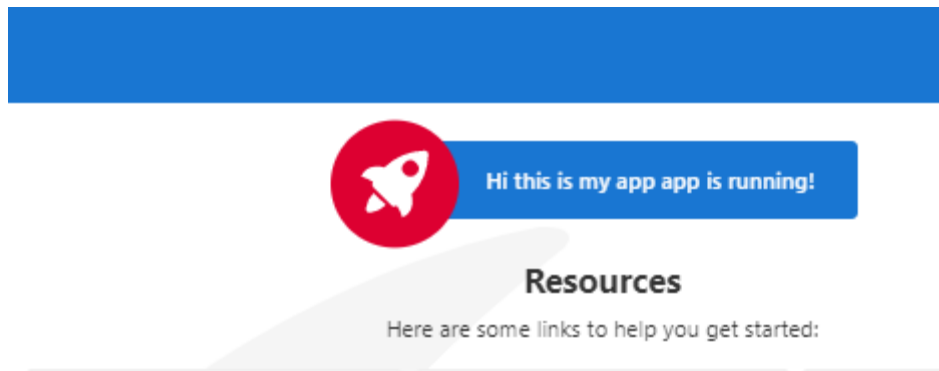
```
src > app > TS app.component.ts >  AppComponent
```

```
1  import { Component } from '@angular/core'
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'my app';
10 }
11
```

```
<h1>
  Hi, this: {{ title }}!
</h1>
```

# Editing the First App

- Now interestingly, we right-click on the loaded page and click “View page source” we don’t actually see that code there.



# Editing the First App

- Now interestingly, we right-click on the loaded page and click “View page source” we don’t actually see that code there. We just see a bunch of script imports at the bottom. That is our build code and the Angular framework code, the head tag here and then this strange app-root part. Now, app-root is also something we see in the app.component.ts file here in the selector.



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>MyDreamApp</title>
6   <base href="/">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11   <app-root></app-root>
12 <script src="runtime.js" type="module"></script><script src="polyfills.js" type="module"></script><script src="styles.js" type="module"></script><script src="vendor.js" type="module"></script><script src="main.js" type="module"></script></body>
13 </html>
14
```

# Editing the First App

- Now

```
import { Component } from '@angular/core';

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

The "selector"

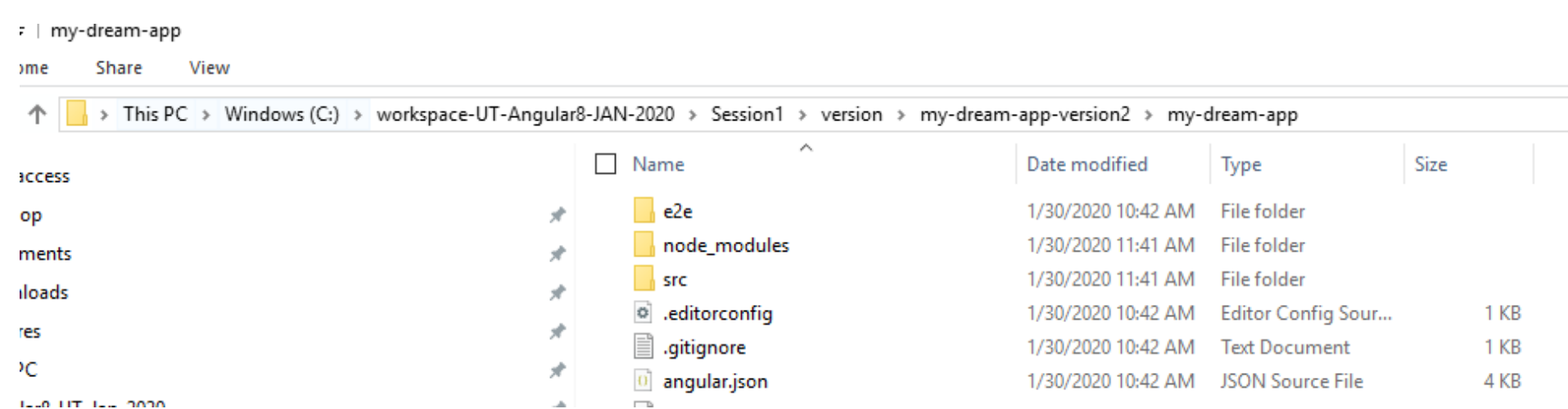
app.component.ts file

# Editing the First App

- Create following directory:
- C:\workspace-UT-Angular8-SEPT-2020  
  \Session1\versions\my-dream-app-version2
- Copy “my-dream-app” to “C:\workspace-UT-Angular8-SEPT-2020 \Session1\versions\my-dream-app-version2 “

# Editing the First App

- Closed Visual Studio Code
- Right click inside directory C:\workspace-UT-Angular8-SEPT-2020\Session1\versions\my-dream-app-version2\my-dream-app and choose “Open with Code”.

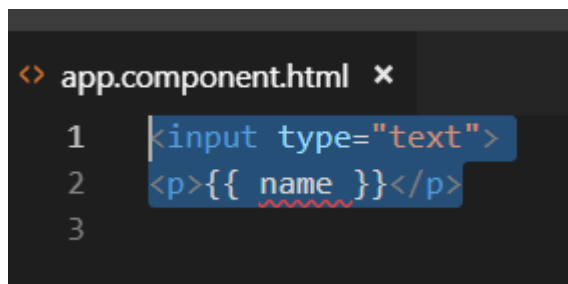


# Editing the First App

- Delete contents in C:\workspace-UT-Angular8-SEPT-2020\Session1\versions\my-dream-app-version2\my-dream-app\src\app\app.component.html.
- The contents should be:

`<input type="text">`

`<p>{{name}}</p>`

A screenshot of a code editor window titled 'app.component.html'. The editor shows three lines of code: line 1 is '<input type="text">', line 2 is '<p>{{ name }}</p>', and line 3 is empty. The code is syntax-highlighted, with the opening and closing tags in blue and the attribute values in orange. The variable name 'name' in the interpolation is underlined with a red squiggly line, indicating it is not defined. The editor has a dark background and a light-colored border.

```
<> app.component.html x
1  <input type="text">
2  <p>{{ name }}</p>
3
```

# Editing the First App

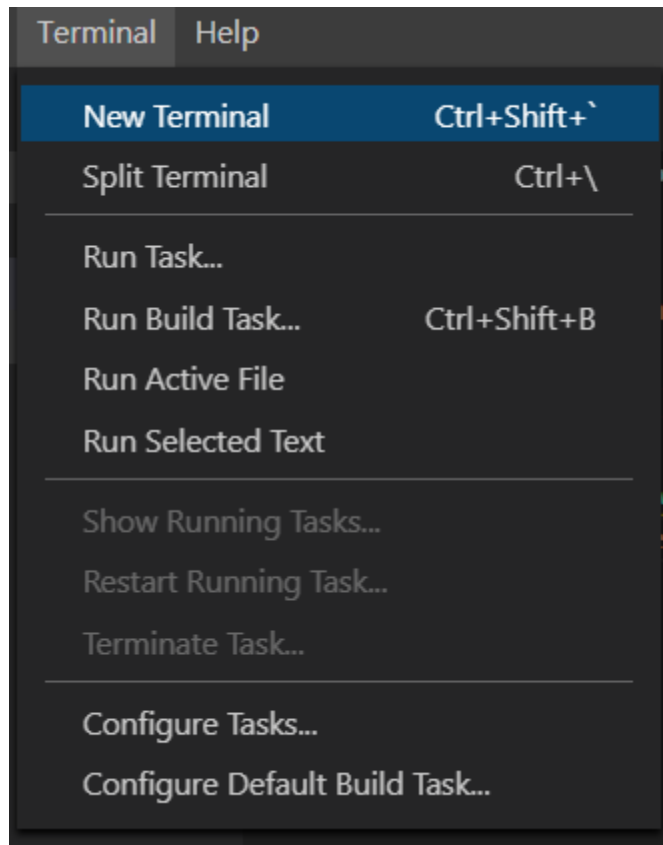
- Go to C:\workspace-UT-Angular8-SEPT-2020  
\Session1\versions\my-dream-app-version2\my-dream-app\src\app\app.component.ts

```
app.component.ts - my-drea
> app.component.html TS app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    name = 'Albert Lam';
10 }
11
```



# Editing the First App

- Go to Terminal/New Terminal:

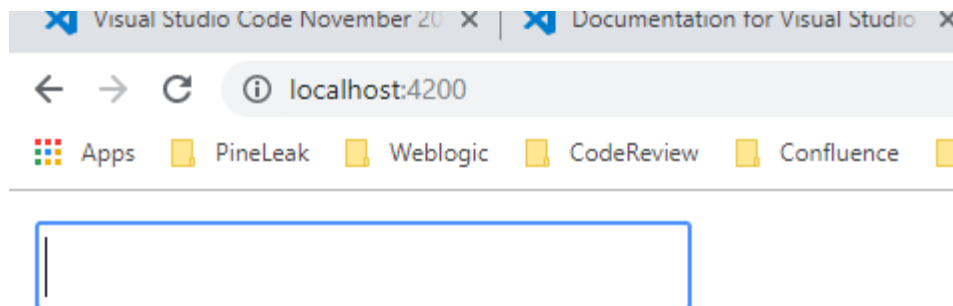


# Editing the First App

- In terminal type “ng serve”.

```
C:\workspace-UT-Angular7-Sept-2019\Session1\versions\my-dream-app-version2\my-dream-app>ng serve
```

Go to browser “localhost:4200”. We see the input and we see the name.



Albert Lam

# Editing the First App

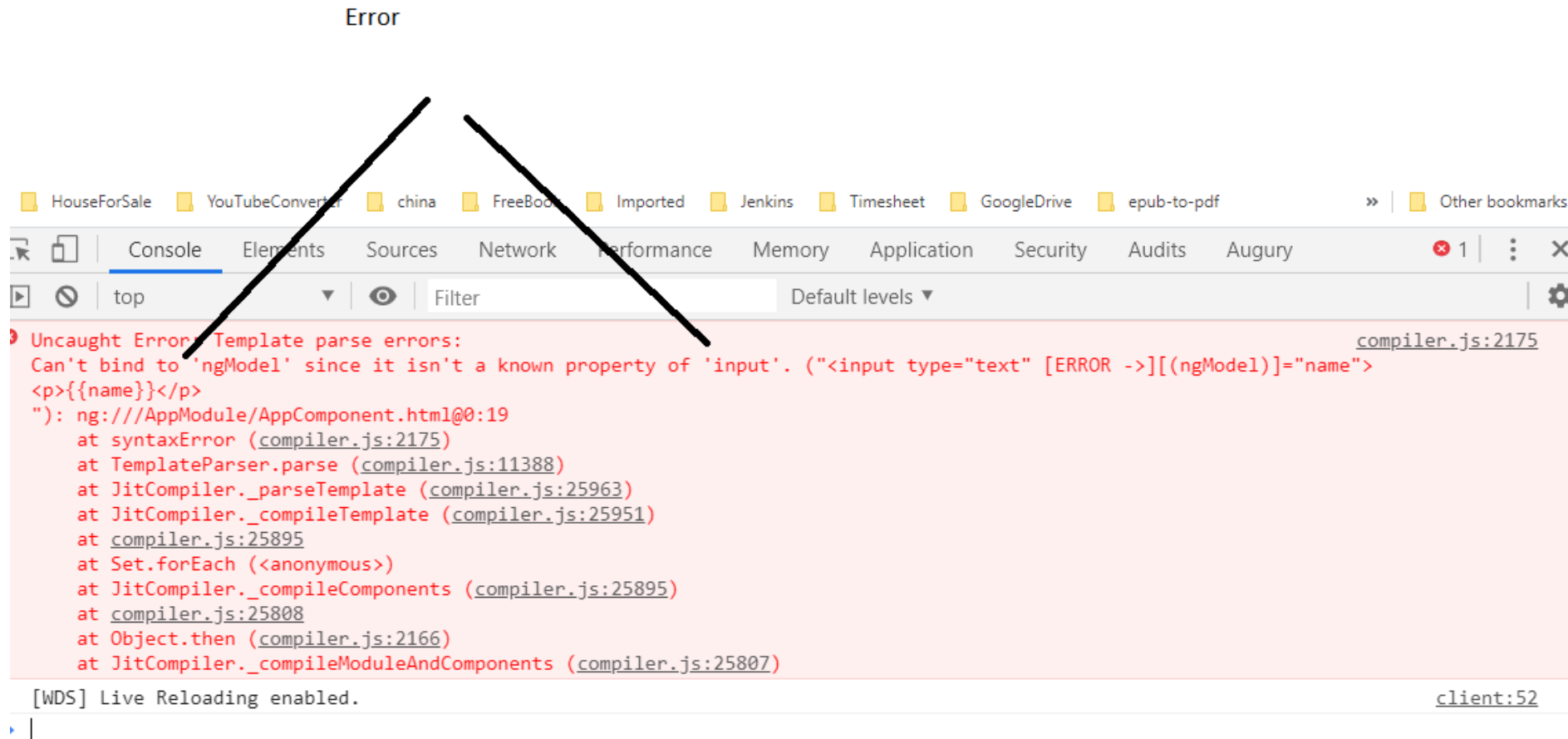
- Now I want to be able to enter something in the input and automatically change the name. We can do this with a tool provided by Angular; a so-called directive which is called ngModel. Now, you'll learn the exact syntax throughout the course. For now, let's simply add [(ngModel)], which like that. Make sure to get the casing correct. So ngModel in square brackets and parentheses on the input and set it equal to name. Now this is a so-called directive and what it does is it basically tells Angular to listen to anything you enter here and store it in this name property, in this name model, but also on the other hand, output the value of the name model in this input.
- App.component.html:

```
<input type="text" [(ngModel)]= "name">  
<p>{{name}}</p>
```

```
<input type="text" [(ngModel)]= "name">  
<p>{{ name }}</p>
```

# Editing the First App

- This is what we're doing here. Now, if we save this we don't see anything on the page and if we open the developer tools in google chrome and can see it under console. Can't bind to 'ngModel' since it isn't a known property of 'input'.



# Editing the First App

- So somehow Angular doesn't understand ngModel.
- Now that's strange, because as I said it's built-in right?
- Angular is actually split up into multiple modules; sub-packages you could say. We need to add them if you want to use a certain feature from them. To add such a feature, we go to another file that we haven't had a look at yet; the app.module.ts file. This is basically where we tell Angular which pieces belong to our app and there we have to add something to imports to import another package from Angular. So we need to import it at the top of the file first because Typescript always needs to know where things are. So, `import { FormsModule } from '@angular/forms';`

# Editing the First App

- So somehow Angular doesn't understand ngModel.
- Now that's strange, because as I said it's built-in right?

EXPLORER

Warn Guides extension has detected that you are using "editor.renderIndentGuides" settings. Guides will no...

OPEN E... 1 UNSAVED

Welcome

app.component.html src\app

User Settings C:\Users\tuyet\..

TS app.component.ts src\app

TS app.module.ts src\app

MY-DREAM-APP

e2e

node\_modules

src

app

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

assets

environments

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 import { FormsModule } from '@angular/forms';
7
8 @NgModule({
9   declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     FormsModule
15   ],
16   providers: [],
17   bootstrap: [AppComponent]
18 })
19 export class AppModule { }
20
```

New Code

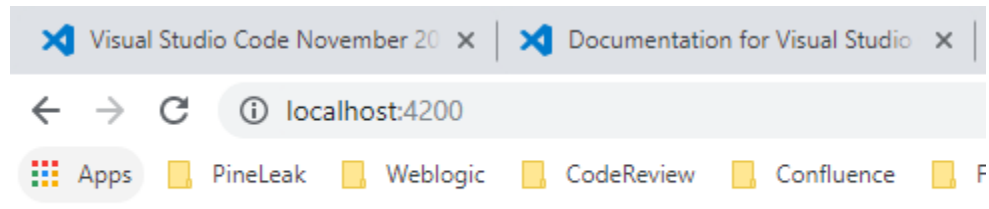
# Editing the First App

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';  
import { FormsModule } from '@angular/forms';
```

```
@NgModule({  
  declarations: [  
    AppComponent  
  ],  
  imports: [  
    BrowserModule,  
    FormsModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

# Editing the First App



Albert Lam2222

Albert Lam2222



# TypeScript

- What is typescript.
- Typescript, really is just a superset of javascript.
- It offers more features than vanilla JavaScript, like classes, interfaces and, very important, which gives it the name, types; strong typing.
- So, you define in TypeScript if a certain variable is a number or a string or something else. You don't do this in vanilla JavaScript. There, you have dynamic typing. You can have a String variable and then you can assign a number and that's totally fine.
- That won't work in Typescript. It will give you an error and therefore it allows you to write more robust code which gets checked at the time you write it; and not just at the time you run it. This is a great enhancement. However, Typescript doesn't run in the browser, so it is compiled to JavaScript in the end.

# TypeScript

- This compilation is handled by the CLI; one of the reasons why we need the CLI, why we need a CLI, we need a project management tool like the CLI.
- This compilation is really vast, therefore in the browser Javascript is going to run. We're not writing the Angular app in Javascript though, while technically possible that won't be much fun. A lot of the features really only exist in Typescript and Angular is meant to be used with TypeScript.

## **10. A Basic Project Setup using Bootstrap for Styling**

- Created a directory C:\workspace-UT-Angular8-SEPT-2020\versions\my-dream-app-version3-bootstrap
- Copied C:\workspace-UT-Angular8-SEPT-2020\Session1\versions\my-dream-app-version2\my-dream-app to C:\workspace-UT-Angular8-SEPT-2020\Session1\versions\my-dream-app-version3\
- Right click on folder C:\workspace-UT-Angular8-SEPT-2020\Session1\versions\my-dream-app-version3-bootstrap\my-dream-app and choose “open with code”.

## 10. A Basic Project Setup using Bootstrap for Styling

- Step 1:
- Run the following command to install bootstrap locally in Microsoft Studio Terminal or Mac Terminal or Dos-prompt:  
`npm install --save bootstrap@3`

```
C:\workspace-UT-Angular8-JAN-2020\Session1\version\my-dream-app-version3-bootstrap\my-dream-app>npm install --save bootstrap@3
[1] \ loadExtraneous: sill resolveWithNewModule bootstrap@3.4.1 checking installable status
```

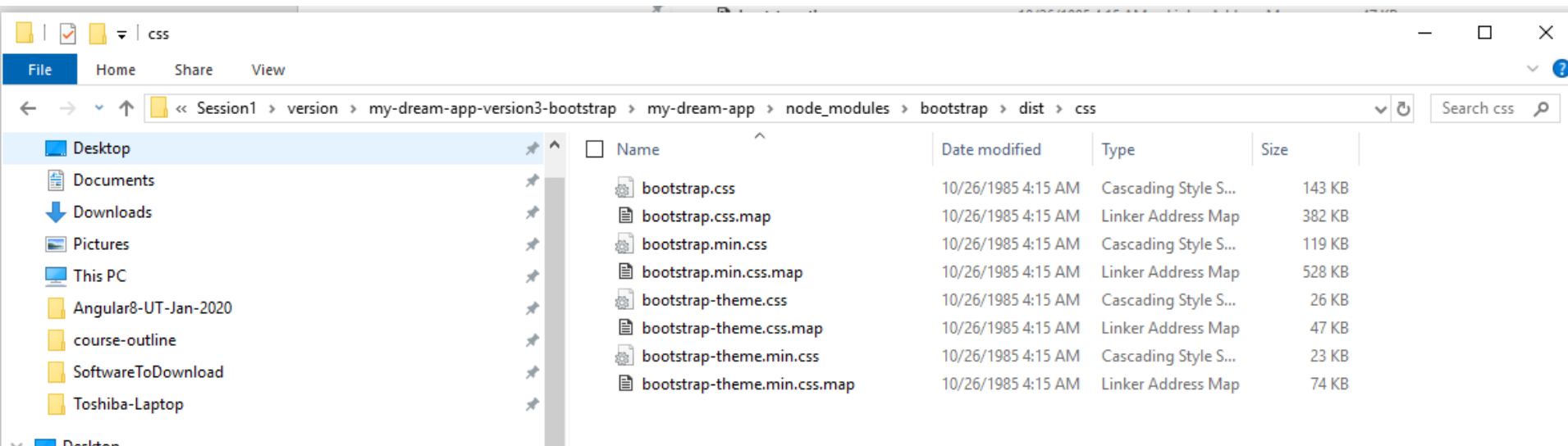
# 10. A Basic Project Setup using Bootstrap for Styling

- Step 2:
- Under node\_modules:

└─ bootstrap	6
└─ dist	7
└─ css	8
# bootstrap-theme.css	9
# bootstrap-theme.css.map	10
# bootstrap-theme.min.css	11
# bootstrap-theme.min.css.map	12
# bootstrap.css	13
# bootstrap.css.map	14
# bootstrap.min.css	15
# bootstrap.min.css.map	16
▸ fonts	17
▸ js	18
▸ fonts	19
	20
	21

# 10. A Basic Project Setup using Bootstrap for Styling

- Step 2: Verify location of css file.
- Under node\_modules:



## 10. A Basic Project Setup using Bootstrap for Styling

- Step 3:
- Open the following file: C:\workspace-UT-Angular8-SEPT-2020\Session1\versions\my-dream-app-version3-bootstrap\my-dream-app\angular.json
- Original:

```
"styles": [  
  "src/styles.css"  
],
```

- New Modified:

```
"node_modules/bootstrap/dist/css/bootstrap.min.css",  
"src/styles.css"
```

# 10. A Basic Project Setup using Bootstrap for Styling

- Step 3:

```
"projects": {
  "my-dream-app": {
    "root": "",
    "sourceRoot": "src",
    "projectType": "application",
    "prefix": "app",
    "schematics": {},
    "architect": {
      "build": {
        "builder": "@angular-devkit/build-angular:browser",
        "options": {
          "outputPath": "dist/my-dream-app",
          "index": "src/index.html",
          "main": "src/main.ts",
          "polyfills": "src/polyfills.ts",
          "tsConfig": "src/tsconfig.app.json",
          "assets": [
            "src/favicon.ico",
            "src/assets"
          ],
          "styles": [
            "node_modules/bootstrap/dist/css/bootstrap.min.css",
            "src/styles.css"
          ],
          "scripts": [],
          "es5BrowserSupport": true
        },
        "defaultConfiguration": ""
      },
      "serve": {
        "builder": "@angular-devkit/build-angular:browser",
        "options": {
          "outputPath": "dist/my-dream-app",
          "index": "src/index.html",
          "main": "src/main.ts",
          "polyfills": "src/polyfills.ts",
          "tsConfig": "src/tsconfig.app.json",
          "assets": [
            "src/favicon.ico",
            "src/assets"
          ],
          "styles": [
            "node_modules/bootstrap/dist/css/bootstrap.min.css",
            "src/styles.css"
          ],
          "scripts": [],
          "es5BrowserSupport": true
        },
        "defaultConfiguration": ""
      },
      "extract-i18n": {
        "builder": "@angular-devkit/build-angular:extract-i18n",
        "options": {
          "outputPath": "dist/my-dream-app"
        }
      }
    }
  }
}
```

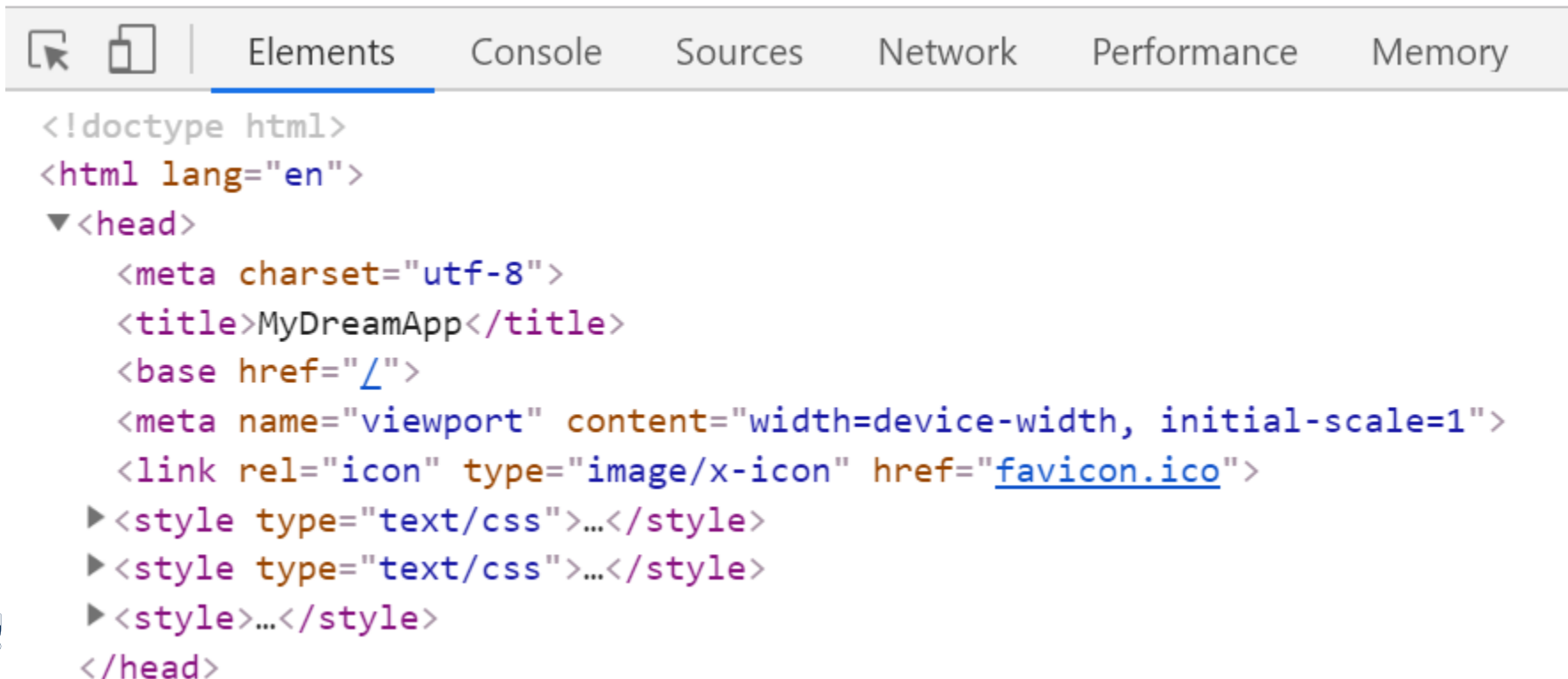
Add this line





## 10. A Basic Project Setup using Bootstrap for Styling

- Step 4:
- Run the following command: `ng serve`
- Step 5:
- Click the first style in head under elements in google chrome:



The screenshot shows the Google Chrome DevTools 'Elements' panel. The 'head' section is expanded, displaying the following HTML code:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>MyDreamApp</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <style type="text/css">...</style>
    <style type="text/css">...</style>
    <style>...</style>
  </head>
```



# 10. A Basic Project Setup using Bootstrap for

```
<!doctype html>
<html lang="en">
...▼<head> == $0
  <meta charset="utf-8">
  <title>MyDreamApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  ▼<style type="text/css">
    /*!
     * Bootstrap v3.4.1 (https://getbootstrap.com/)
     * Copyright 2011-2019 Twitter, Inc.
     * Licensed under MIT
     (https://github.com/twbs/bootstrap/blob/master/LICENSE)
     *//*! normalize.css v3.0.3 | MIT License |
     github.com/necolas/normalize.css */html{font-family:sans-serif;-ms-
     text-size-adjust:100%;-webkit-text-size-
     adjust:100%}body{margin:0}article,aside,details,figcaption,figure,fo
     oter,header,hgroup,main,menu,nav,section,summary{display:block}audio
     ,canvas,progress,video{display:inline-block;vertical-
```