

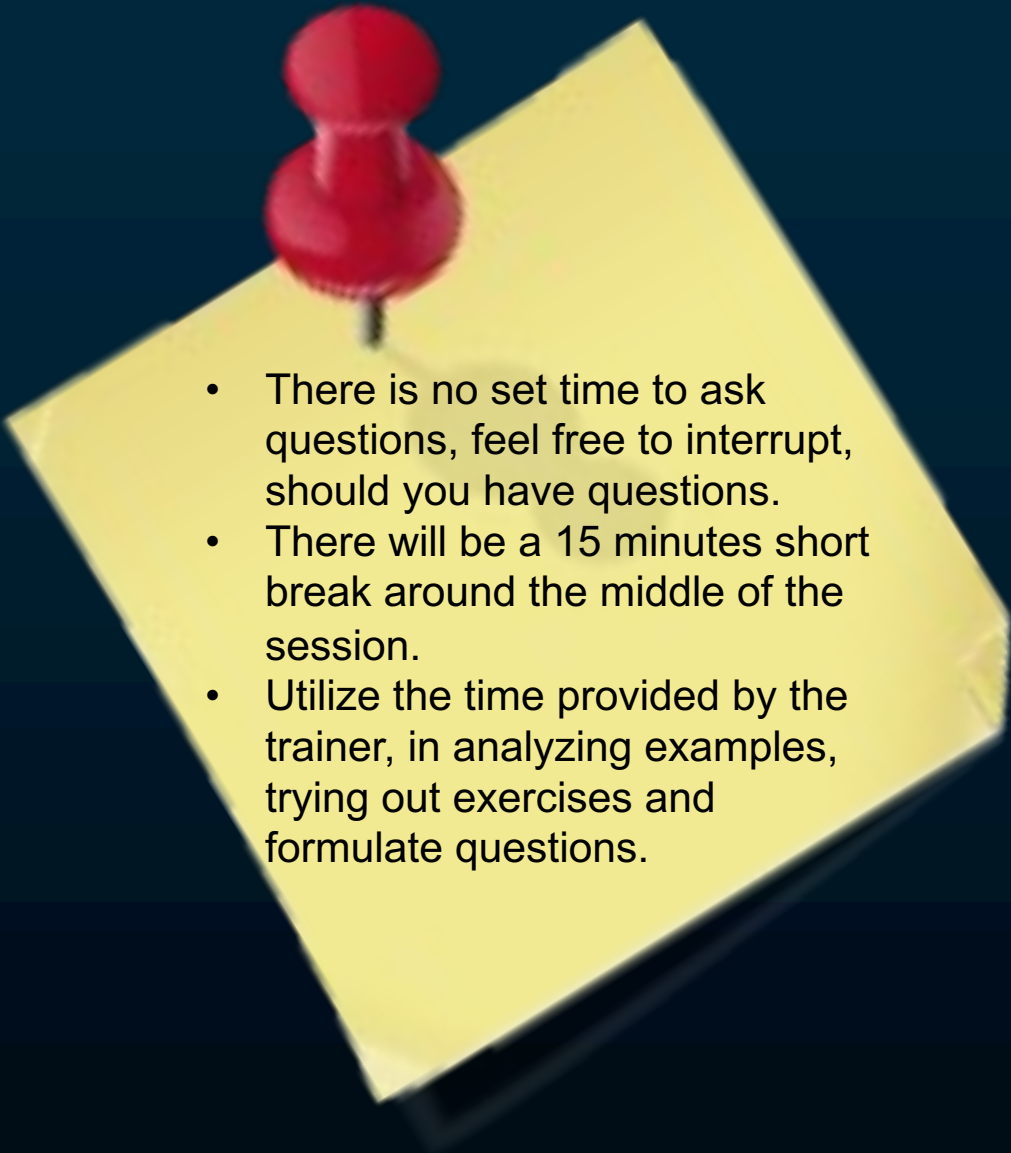
Introduction to Angular

Vignesh Murali Natarajan - 119780

Jason Monroe – 688776

Taryn Ernd - 785108

Session Rules

- 
- There is no set time to ask questions, feel free to interrupt, should you have questions.
 - There will be a 15 minutes short break around the middle of the session.
 - Utilize the time provided by the trainer, in analyzing examples, trying out exercises and formulate questions.

Course Objective:

At the end this session, you will be able to

- Set up an Angular workspace and create a new Angular application
- List the files in an Angular application involved in the bootstrapping process

Key Topics

Intro

- SPA, RWD
- Angular Versions
- Landscape
- Hello World

Basics

- Components
- Directives
- Routing
- Services
- HttpClient

Advanced

- Defining Observables
- Route Guards
- Material
- Pipes



Objectives

After completing this session, you will be able to comprehend the following topics:

- Introduction to SPA
- Workspace Setup
- Introduction to Angular
 - Angular JS vs Angular
 - High Level Architecture
- Hello Angular !!



Objectives

- Walk Through Angular Project

angular.json

index.html

main.ts

app.module.ts

app.component.ts

app.component.html

Assets folder



Introduction to SPA

Single Page Applications

- Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page, as the user interacts with the app.
- SPAs use AJAX and HTML5 to create fluid and responsive Web apps, without constant page reloads. This means much of the work happens on the client side, in JavaScript.

Single Page Applications (Contd.)

- Web app that fits on a single web page is Fluid UX, similar to a desktop app. Few examples would be Gmail, Google maps and so on
- Html page contains mini---views (HTML Fragments) that can be loaded in the background
- No reloading of the page, hence better User Experience
- Requires handling of browser history, navigation and bookmarks

Responsive Web Design

- Responsive Web Design (RWD) is an approach to web design, aimed at crafting sites to provide an optimal viewing experience, easy reading and navigation with a minimum of resizing, panning, and scrolling, across a wide range of devices (from desktop, computer monitors to mobile phones).
- A responsive theme's layout adjusts to your device's screen size. It provides an optimal viewing experience, easy reading and navigation on mobile phones, tablets, laptop and desktop computers.

Responsive Web Design

Rise of RWD and SPA



Challenges in SPA

- DOM Manipulation: How to manipulate them effectively
- History: What happens at the push of the browser back button
- Routing: Readable URLs?
- Data Binding: How to Bind Data from Model to View
- View Loading: How to load the view
- Lot of coding! A framework to the rescue

Workspace Setup

Required Software

- Required Software:
 - Node-v10.x <https://nodejs.org/en/download/>
 - Visual Studio Code
- Additional Setup:
 - After the installation of node, ensure that you install the following, using Node Package Manager(npm):
 - Angular CLI

```
$ npm install -g @angular/cli
```

Verify Version with
ng --version

Angular @cli

Angular CLI: 8.3.12
Node: 12.13.0
OS: darwin x64
Angular:
...

Package	Version
@angular-devkit/architect	0.803.12
@angular-devkit/core	8.3.12
@angular-devkit/schematics	8.3.12
@schematics/angular	8.3.12
@schematics/update	0.803.12
rxjs	6.4.0

Introduction to Angular

Angular

- Angular 2+ (commonly referred to as 'Angular') is a Typescript-based open-source front-end web application platform, led by the Angular Team at Google.
- One way to build applications and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile, and native desktop.
- Achieve the maximum speed possible on the Web Platform and take it further, through Web Workers and server-side rendering.
- Angular is a complete rewrite from the same team that built AngularJS.

Angular Landscape

- Strong Typing
- Decorators (Annotations)

- Classes
- Modules
- Arrow Functions

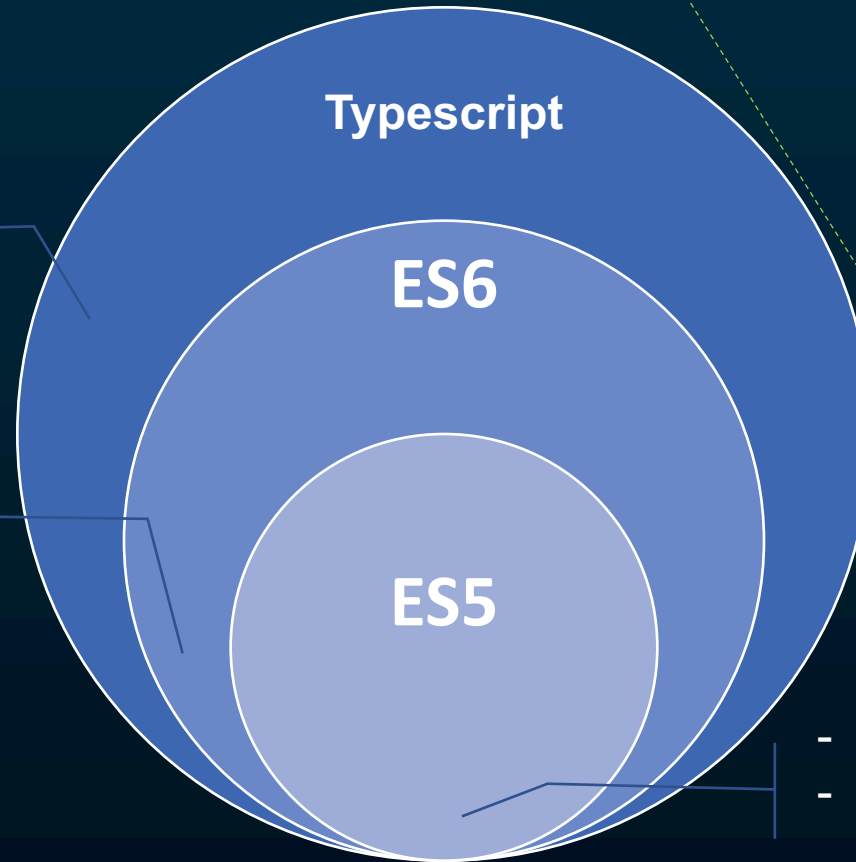
Flow

Typescript

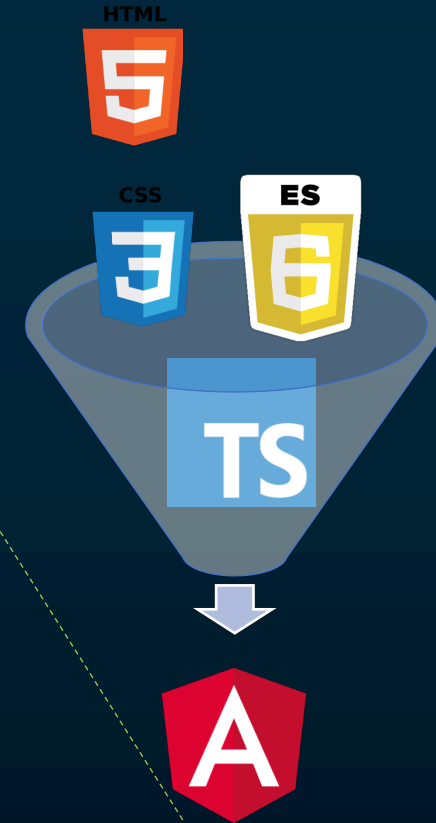


Transpile

Javascript



- Target
- Needs Shim to execute ES6



Why Angular?

- Distilled all the best practices of Angular 1.x into Angular 7 (See Appendix 1 in this Deck)
- By focusing on standards, we get **twice** the power with **half** the framework
- Reactive mechanisms baked into the framework like two-way binding
- The tooling support is as good as on Java or .Net platforms
- Using TypeScript classes and interfaces makes the code more concise and easy to read and write

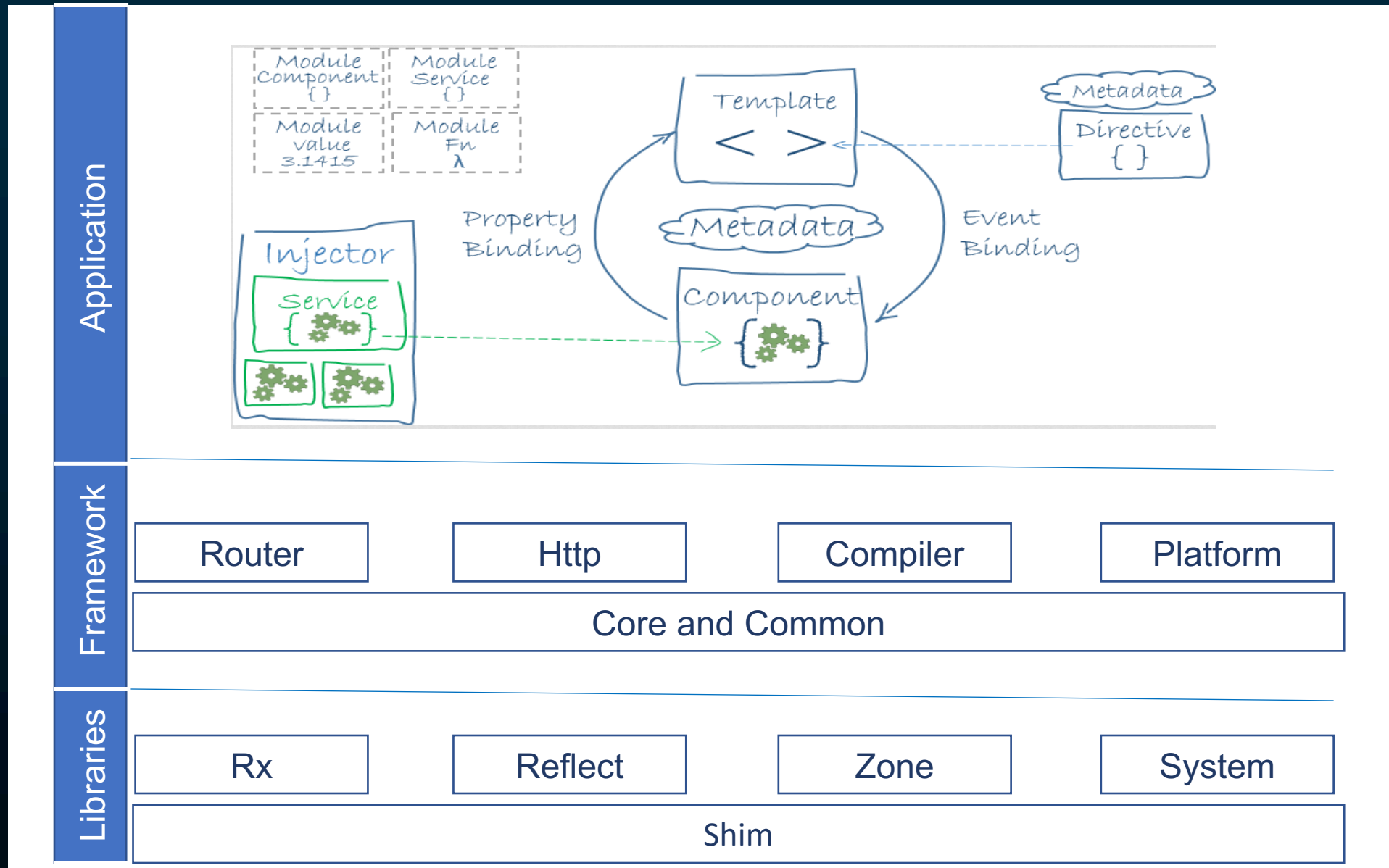
Why Angular?

- Clean separation between the code that renders UI and the code that implements application logic
- The UI does not have to be rendered in HTML. There are already products, supporting native UI rendering for iOS and Android
- The TypeScript compiler generates JavaScript that a human can read and compile into ES3, ES5, or ES6 versions of JavaScript

Why Angular?

- Dependency injection gives you a clean way to implement loose coupling between components and services.
- Automatic (and fast) change detection mechanism spares you from the need to manually force UI updates, while giving you a way to fine-tune this process.
- Angular comes with the Rx.js library, which allows you to arrange a subscription-based processing of asynchronous data and eliminates the callback hell.
- An ability to pre-compile the code eliminates the need to package Angular Module Loader (not to be confused with TypeScript compiler) with your app, which further minimizes the overhead of the framework.
- The scaffolding and deployment tool (Angular CLI) spares developers from writing the boilerplate code and configuration scripts.

Angular Architecture



Hello Angular

Hello Angular – NG Way

- To create a new project, from a terminal window, go to the folder where you want to create the project and issue the command below:

```
$ ng new <<your-project-name-here>>
```

- Base project files creation and dependency resolution will be done by Angular CLI for you
- When the above command completes successfully, launch the Angular application by running the command

```
$ ng serve --o
```

- Launches the index.html in a browser @ <http://localhost:4200> (port might vary)
- Eureka!!

Angular Deployment

<https://angular.io/guide/deployment>

- Ng serve
 - In-memory compilation running on local server
 - File changes trigger reload for iterative development
- Ng build
 - Creates output folder 'dist'
 - Consolidates app into a few js files
 - Assets folder replicated
 - Must change base href for deployment environment

Hello Angular – NPM Way

- Manually create a folder on you dev machine, navigate to it in a terminal, then create the angular substrate by running the below command:

```
$ npm init -y
```

- Inspect the files and folders generated, particularly the package.json file. Only the name, description and scripts attributes are of significance in that file as of now.
- Add dependencies – Angular, SystemJS, Live-server, TypeScript compiler
- Run the install command to install the above dependencies

```
$ npm install
```

Hello Angular – NPM Way

- Copy the following files - SystemJS Config file, index.html, main.ts, app.module.ts and app.component.ts
- Launch the Angular application by running the command

```
$ npm start
```
- Launch the index.html in a browser @ <http://localhost:4200/index.html> (port might vary)
- Eureka!!

Angular Project Structure

angular.json

```
angular.json x
1 {
2   "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
3   "version": 1,
4   "newProjectRoot": "projects",
5   "projects": {
6     "Hello-Angular-Complete": {
7       "root": "",
8       "sourceRoot": "src",
9       "projectType": "application",
10      "prefix": "app",
11      "schematics": {},
12      "architect": {
13        "build": {
14          "builder": "@angular-devkit/build-angular:browser",
15          "options": {
16            "outputPath": "dist/Hello-Angular-Complete",
17            "index": "src/index.html",
18            "main": "src/main.ts",
19            "polyfills": "src/polyfills.ts",
20            "tsConfig": "src/tsconfig.app.json",
21            "assets": [
22              "src/favicon.ico",
23              "src/assets"
24            ],
25            "styles": [
26              "src/styles.css"
```

- “**outputPath**” is the location where the ng build command places the condensed js files for the entire app
- **index.html** is the single page in which the Angular App is injected
- **main.ts** is the “main” Angular entry point
- **polyfills.ts** contains browser compatibility polyfills and application polyfills.
- **styles.css** is where global styles can be placed. Any CSS rules you place here are injected into the DOM in a `<style></style>` tag
- “**assets**” array describes the location of static file assets

index.html

```
<> index.html x
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>HelloAngularComplete</title>
6      <base href="/">
7
8      <meta name="viewport" content="width=device-width, initial-scale=1">
9      <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12     <app-root></app-root>
13 </body>
14 </html>
15
```

Angular content is injected in the <app-root></app-root> tag

main.ts

```
TS main.ts ×
1  import { enableProdMode } from '@angular/core';
2  import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4  import { AppModule } from './app/app.module';
5  import { environment } from './environments/environment';
6
7  if (environment.production) {
8    | enableProdMode();
9  }
10
11  platformBrowserDynamic().bootstrapModule(AppModule)
12  | .catch(err => console.error(err));
13
14
```

AppModule is a TypeScript class (imported from /app/app.module) that is being bootstrapped on line 11 as the Angular start-up module.

app.module.ts

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

AppComponent is a TypeScript class (imported from /app/app.component) that is being bootstrapped on line 14 as the initial component for the module.

Bootstrapping

- Bootstrapping is an essential process in Angular 5 where the application is loaded when Angular comes to life.
- The key files in an Angular application include -
 - ***app/app.component.ts*** - Root component is defined here.
 - ***app/app.module.ts*** - Entry Angular Module to be bootstrapped is defined here.
 - ***index.html*** - Page the components are rendered in.
 - ***app/main.ts*** - Glue that combines the component and page.
- The bootstrap process loads ***main.ts*** which is the main entry point of the application. The bootstrap code should be confined to the main.js file.
- Angular is not a web-only based framework. Components that can run in NativeScript, Cordova etc can also be written.
- Bootstrapping is a process interprets imports based on the application platform and the environment.

app.component.ts

```
TS app.component.ts ✕
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 = 'Hello-Angular-Complete';
10 }
```

@Component decorator function selects app-root element in index.html and injects into it the app.component.html combined with the TypeScript in the AppComponent class and app.component.css

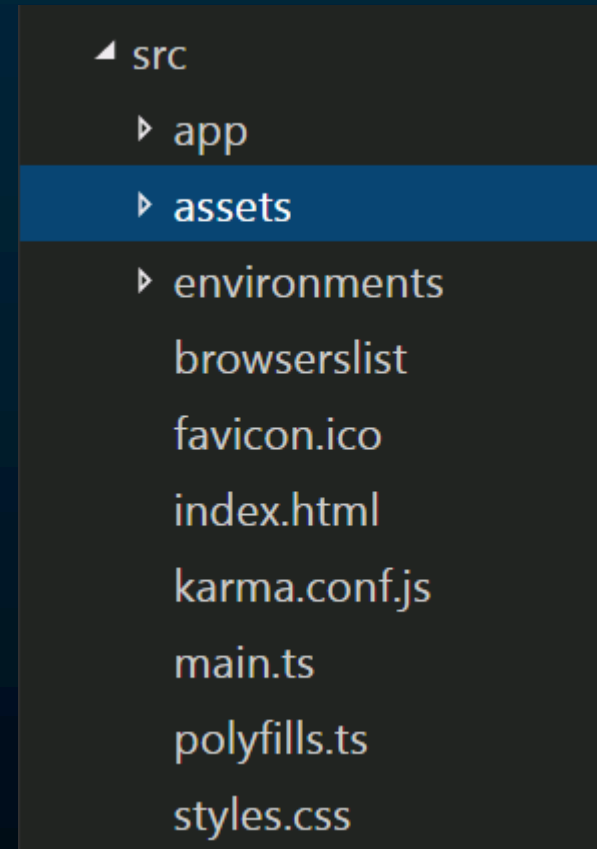
app.component.html

```
<> app.component.html x
1  <!--The content below is only a placeholder and can be replaced.-->
2  <div style="text-align:center">
3    <h1>
4      Welcome to {{ title }}!
5    </h1>
6    Tour of Heroes</a></h2>
12   </li>
13   <li>
14     <h2><a target="_blank" rel="noopener" href="https://github.com/angular/angular-cli/wiki">CLI Documentation</a></h2>
15   </li>
16   <li>
17     <h2><a target="_blank" rel="noopener" href="https://blog.angular.io/">Angular blog</a></h2>
18   </li>
19 </ul>
20
21
```

Double curly brace syntax on line 4 instructs Angular to look for a variable named 'title' in AppComponent and place its value in the DOM.

Assets folder

- Replicated when Angular project is built
- Place any static files here
- Can use relative references



Review

- 1.The SPA / RWD Landscape
- 2.What is Angular and the distinction between versions
- 3.Why use Angular
- 4.How to build a simple Angular App
- 5.What are the major components in an Angular App



Course Objective:

At the end this session, you will be able to

- Set up an Angular workspace and create a new Angular application
- List the files in an Angular application involved in the bootstrapping process

What are the advantages and disadvantages pushing the templating to the Browser?

Angular 8 vs Angular 1

Angular JS	Angular 7
Uses and built on Javascript. EcmaScript 5	Used TypeScript, a superset of Javascript. Also supports EcmaScript 6
No Mobile Support	Mobile Oriented, completely decoupled MVVM
\$Scope plays a pivotal role	No \$Scope. Change Detection is done using Zone.js
No Type support, compilation	Provides Type support and AOT Compilation like other programming language
Controllers ruled the show	No Controllers. Components + Directives compensate instead

Angular 8 vs Angular 1 (Contd.)

```
////Angular 1.x using Controller and
$scope.....
var myApp = angular
.module("myModule", [])
.controller("productController",
function($scope) {
    var prods = { name:
"Prod1", quantity: 1 };
    $scope.products = prods;
});
```

```
///Angular 2 Components using TypeScript.....
import { Component } from 'angular2/core';
@Component({
  selector: 'prodsdata',
  template: `
<h3>{{techncalDiary}}</h3> `
})
export class ProductComponent {
  prods = { name: 'Prod1', quantity: 1 };
}
```

Source

- [Angular API Documentation](#)
- [Angular.io](#)
- [Front End Masters](#)
- [Rangle.io](#)



Disclaimer: Parts of the content of this course is based on the materials available from the Web sites and books listed above. The materials that can be accessed from linked sites are not maintained by Cognizant Academy and we are not responsible for the contents thereof. All trademarks, service marks, and trade names in this course are the marks of the respective owner(s).

Thank You

