Angular 6 Training Course

Exercise A-intro

- This exercise builds a **minimal app** using the **Angular CLI** command line tools.

Setup

- *The course examples assume you have completed the setup instructions.*
- Open the command-line.
- On **Mac** open the **terminal**, on **PC** open the **command-prompt**.
- Change directory to the desktop

```
cd desktop
```

- Run **ng new** to create a new Angular project.

```
ng new hello
```

- Change directory into this folder

```
cd hello
```

- Run the ng serve command shown below.
- This will run **Webpack** to **transpile** and bundle your ES6 and Typescript code.
- The Angular CLI will create a local webserver, and open the generated files in a browser at **localhost:4200**.
- A web page with the Angular icon should appear.

```
ng serve --open
```

Understanding the structure of an Angular app.

- Open the **/hello** folder in Atom.
- The **Angular CLI** has created a hierarchy of files/folders.
- The file **.angular-cli.json** defines the location/names of key entry-point files.
- These include **src/index.html** and **src/main.ts**

index.html, main.ts

- Index.html contains an instance of the **top-level component**.

```
<app-root></app-root>
```

- **main.ts** contains some standard boilerplate Angular code to initialise your application.
- It refers to the **top-level module** AppModule, which contains your components.

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

AppModule

- AppModule is defined in **src/app/app.module.ts**
- It imports/loads the top-level component **AppComponent**.

```
@NgModule({
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
```

AppComponent

- Component AppComponent is defined in **src/app/app.component.ts**
- It defines an ES6 class that is **wrapped/decorated** in Angular metadata.
- Angular components use a **decorator** pattern/annotation.
- This defines the CSS styles, HTML template and custom-tag associated with this class.

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

Edit the main component

- If we change the **component selector**

```
selector: 'weather'
```

- We also need to change the component instance in **index.html**

```
<weather></weather>
```

Styles, templates

- We can define local styles and templates within **app.component.ts**

```
    @Component({
      selector: 'weather',
      template: `
        <section class='news'>Hurricane Angular arrives
today</section>
      `,
      styles: [`.news{
          font-size:2.5rem;
          font-family:tahoma
      }`]
    })
```

- If we **rename** the component

```
export class WeatherComponent {}
```

- we need to update **app.module.ts**

```
import { WeatherComponent } from './app.component';

@NgModule({
  declarations: [
    WeatherComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [WeatherComponent]
})
```

```
export class AppModule {}
```

Define a class property and use it in the template

- We can define a class property and initialise it in the constructor.

```
export class WeatherComponent {
    name : string;
    constructor() {
        this.name = "Angular"
    }
}
```

- We can use it in the template using **string interpolation/moustache syntax**.

```
<section class='news'>
    Hurricane {{name}} arrives today
</section>
```

- The finished version of the component:

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

@Component({
  selector: 'weather',
  template: `
    <section class='news'>Hurricane {{name}} arrives
today</section>
  `,
  styles: [`.news{
      font-size:2.5rem;
      font-family:tahoma
  }`]
})

export class WeatherComponent {
    name : string;
    constructor() {
        this.name = "Angular"
    }
}
```

- We have created a small self-contained Angular component.
- A real project will compose together a hierarchy of related components.
- Our example defines template HTML and CSS styles inline in the component file. A real project will separate these into external files.
- The generated folder /hello contains a large folder /node_modules. This does not need to be archived. You can delete this folder. The project can be regenerated by running **npm install** at the command line.
- This example has combined **ES5**, **ES6**, **Typescript** and **Angular** syntax.
- It was build using the **Angular CLI** and **Webpack bundler**.