Angular 12

(Part-1)

by

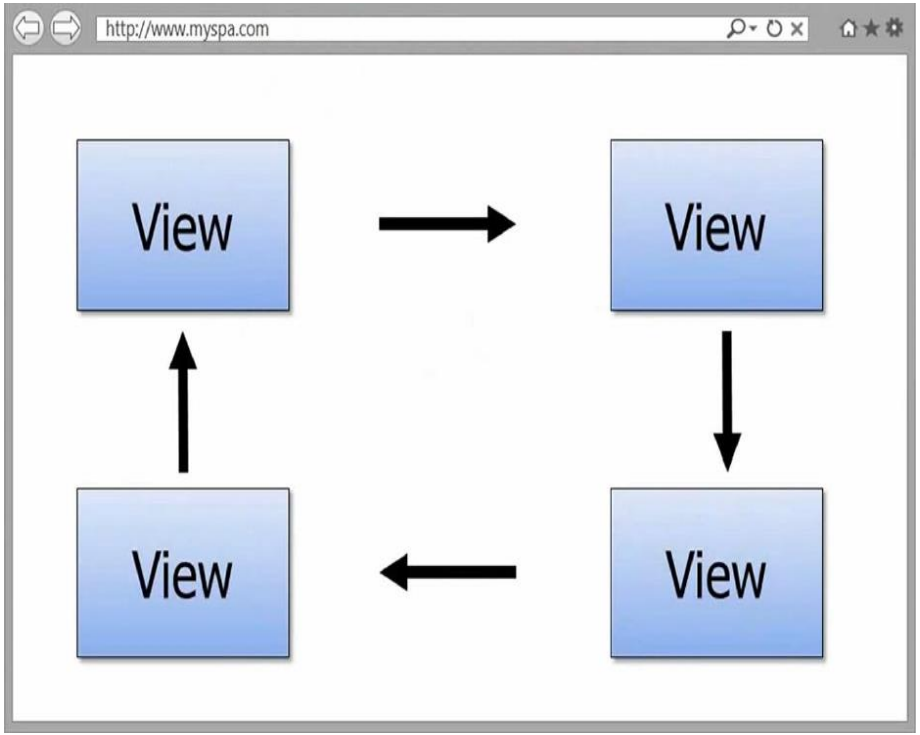
Anand Kulkarni

anand.kulkarni1@zensar.com

Contents

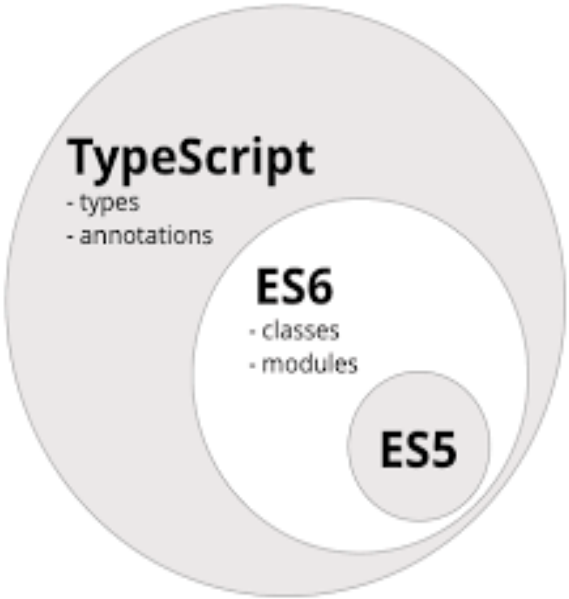
Single Page Application (SPA)

- Single Page Application is one in which we have a shell page and we can load multiple views into that.



TypeScript

- TypeScript is a free and open-source programming language developed and maintained by Microsoft.
- It is a superset of JavaScript, and adds optional static typing and class-based object-oriented programming to the language.
- The major feature supported in TypeScript which is not available in ES6 is ‘types’ & ‘annotations’.
- TypeScript is extensively used by Angular development.
- You can try TypeScript code online at https://www.typescriptlang.org/play/



Angular Installation Setup

- Visual Source Code (https://code.visualstudio.com/download)
- Node JS (https://nodejs.org/en/download/)
- GIT (optional) – (https://git-scm.com/downloads)

Developing Angular Application

1) Change the directory where you wish to create your Angular application.

   cd D:\Anand\Angular

2) Install angular/cli using below command. The angular/cli is a command line interface to build & maintain Angular applications.

   >npm install -g @angular/cli

3) Set the PATH for ‘ng’ command.

   set PATH=%PATH%;c:\Users\ak69498\AppData\Roaming\npm

4) Create a new Angular project named 'hello-world-project ‘:

   >ng new hello-world-project

Developing Angular Application

5) Change the directory to 'hello-world-project' app.

   cd hello-world-project

6) Now, start the angular application:

   >ng serve

7) Open browser & hit http://localhost:4200 , It will show you angular project home page.

Significance of different files in angular app

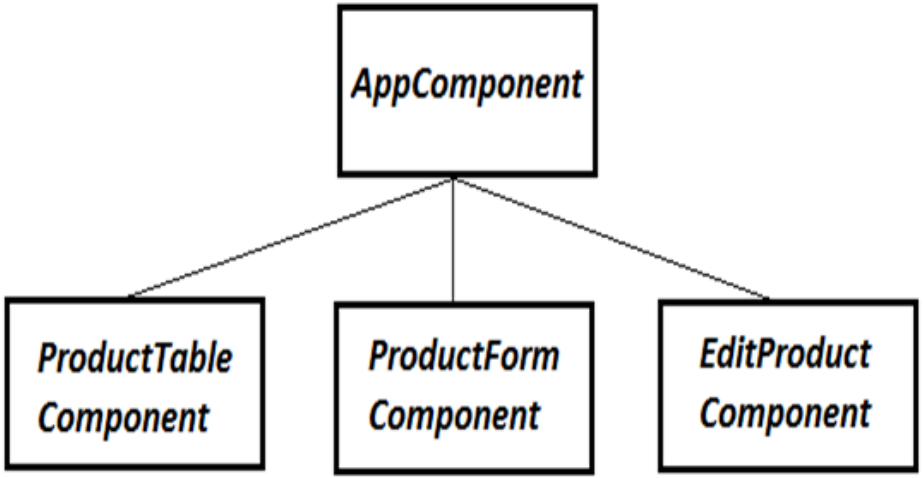
After extracting ‘quickstart-master.zip’, you can see lots of files in ‘quickstart-master’ directory. Let us understand the significance of few important files:

- ***package.json:*** Angular application requires certain libraries or modules for functioning. These are specified in package.json. So when you run ‘npm install’ on command prompt, all dependencies listed in package.json are installed in application’s ‘node_modules’ folder. The package.json also has ‘scripts’ attribute. It configures typescript compiler (tsc) & start lite-server to host your application.
- ***tsconfig.json:*** This file guides typescript compiler to convert typescript code into javascript.

Significance of different files in angular app

- ***style.css:*** It configures styles for out HTML view.
- ***index.html:*** The index.html loads javascripts files, loads ‘app’ module & renders the view using <my-app> tag.
- ***‘main.ts’:*** This is the entry level file of your application.
- ***‘app/app.component.ts’:*** This is the definition of root component.

What is an Angular Application?



Product Application in Angular

- Angular application is nothing but collection of various angular components.
- Angular developers develop various angular components & establish communication among them.

What is an Angular Component?

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

@Component({
        selector: 'my-app',
        template: `<h1>Hello {{name}}</h1>`,
})
export class AppComponent { name = 'Angular App‘ ; }
```

1) A component is nothing but a class with metadata. The class handles properties & business logic actions where as metadata handles the view.
2) Metadata is specified using @Component which is known as decorator.
3) The ‘selector’ is an HTML tag that you will be using to render this component.
4) The ‘template’ is the view rendered when you are loading app component.

How to create an Angular Component?

Angular application is a collection of several components. The app is a root component. Now, we are going to discuss the steps to create a new Angular component:

- Create a folder ‘components’ inside ‘app’. This is optional but recommended.
- Create sample.component.ts in ‘components’ folder with SampleComponent class definition.

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

@Component({
        selector: 'my-sample',
        template: `I am Sample component`
})
export class SampleComponent { }
```

How to create an Angular Component?

- Register sample component into app.module.ts:

```
import { SampleComponent } from './components/sample.component';

@NgModule({

        declarations: [ AppComponent, SampleComponent ],

})
```

- Finally use the newly created SampleComponent inside any other angular component:

```
@Component({
    selector: 'my-app',
    template: `<h1>Hello {{name}}</h1>
    <my-sample></my-sample>`
})
export class AppComponent { name = 'Angular'; }
```

Interpolation & Property Data Binding

- We can declare attributes inside component class. It is also called as properties.
- Component properties can be read using {{attribute}} syntax inside HTML view. It is also called as 'Interpolation'.
- Below is the data flow from component to view a.k.a. one way data binding.

```
@Component({
selector: 'my-app',
template: `<h1>Hello {{name}}</h1>
                        <img [src]="imgLink"/>`
})
export class AppComponent {
        public name = 'Angular';
        public imgLink = 'https://angular.io/assets/images/logos/angular/logo-nav@2x.png';
}
```

Event Binding & References

Event binding helps us to capture the data flow from view to the component.

```
@Component({
        selector: 'my-sample',
        template: `I am Sample component
        <button (click)="onOkClick(myInputBox.value)">OK</button> //Event Binding using ()
        <button (mouseover)="onCancelClick($event)">CANCEL</button> //Event Binding using ()
        <input type='text' #myInputBox value='Angular'/>` //References using #
})
export class SampleComponent {
                onOkClick(value: any) {      console.log("OK clicked: ", value);    }
                onCancelClick(value: any) { console.log("CANCEL clicked: ", value);        }
}
```

Styling components

We can build a component's view more attractive using css styles.

There are three ways to apply styles to the component view:

1) Using 'styles' attribute of @Component decorator.

```
@Component({
        selector: 'my-sample',
        template: `<h4>I am Sample component</h4>
        styles: ['h4 { color: red }']
})
```

Styling components continue…

2) Using ‘styleUrls’ attribute of @Component decorator. The ‘styleUrls’ attribute helps us to create a separate .css file for handling component’s styling.

```
@Component({
        selector: 'my-sample',
        template: `<h4>I am Sample component</h4>
        styleUrls: ‘./sample.component.css’
})
```

3) Adding css style into global css file i.e. styles.css. The styles.css file is created when we create an angular project. Any style added into styles.css will be applicable to all components belong to that application.

CSS Class Binding

You can also apply css classes to angular component:

```
@Component({
   selector: 'my-sample',
   template: `<div [class.myClass]="applyClass">Sample class</div> //CSS class binding
        <div [style.color]="applyColor? 'red' : 'orange'">Sample color</div>` //CSS style binding
})
export class SampleComponent {
  public applyClass = true;
  public applyColor = true
}
```

Thank you!!