**Section02 The Basics**

**Lesson01 How the application works?**

**1-we have to know that the first point the application start is the main.ts**

**2-we have to know that the index.html is the page hosting by the browser**

**import { enableProdMode } from '@angular/core';**

**import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';**

**import { AppModule } from './app/app.module';**

**import { environment } from './environments/environment';**

**if (environment.production) {enableProdMode();}**

**//it will create an instance of the AppModule at the runtime compiler**

**platformBrowserDynamic().bootstrapModule(AppModule)**

**.catch(err => console**.error(err));

**3-on the app.module.ts we type the following code**

**import { BrowserModule } from '@angular/platform-browser';**

**import { NgModule } from '@angular/core';**

**import { AppRoutingModule } from './app-routing.module';**

**import { AppComponent } from './app.component';**

**import { FormsModule } from '@angular/forms';**

**@NgModule({**

**declarations: [AppComponent],**

**imports: [**

**BrowserModule,**

**FormsModule,**

**AppRoutingModule],**

**providers: [],**

**//we see that implement the bootstrap style on the component called AppComponent**

**bootstrap: [AppComponent]})**

**export class AppModule { }**

**4-we have to know that when running the application ng server the angular project generate script files on the bottom section on the html page as below**

**//these script files is responsible to keep track and execute javascript that translated //from the typescript on the web page**

**<body>**

**<app-root \_nghost-akd-c0="" ng-version="8.2.14"></app-root>**

**<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>**

**Lesson02 Component is Important**

**1-Angular in the end is JS framework that change DOM element at the runtime**

**2-component : is key feature in angular that allow to build whole web application from list of component**

**3-each component have its style and its business logic and its html**

**Lesson03 Create New Component**

**Steps: -**

**1-we create folder called server**

**2-we create typescript file called server.component.ts**

**3- we set the following code like below**

**1-Decorator: is Angular feature that enhance elements you are using on the code**

**(which means that notify that this class is type of compoennt)**

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

**@Component({**

**template:'app-server',**

**templateUrl:'./server.component.html'})**

**export class serverComponnet{}**

**Lesson04 understanding the rule of module**

**Notes:-**

**1-we use the component to render on the application**

**2-we use the modules to bundle the component and packing the components**

**3-app.module:is angular feature that are using to bundle the functionality of the app and give the angular which feature does I have to use**

**import { BrowserModule } from '@angular/platform-browser';**

**//we import the NgModule on the app.module**

**import { NgModule } from '@angular/core';**

**import { AppRoutingModule } from './app-routing.module';**

**import { FormsModule } from '@angular/forms';**

**import { AppComponent } from './app.component';**

**import { serverComponnet } from './server/server.component';**

**//we using the @NgModule decorator which have the following 4 sections**

**@NgModule({**

**//using to define the components and pipes and directives**

**declarations: [**

**AppComponent,**

**serverComponnet],**

**//using to define another modules (packages)**

**imports: [**

**BrowserModule,**

**FormsModule,**

**AppRoutingModule],**

**//using to inject services**

**providers: [],**

**//using to provide which component I should bootstrap it on the app.module and //bootstrapped it**

**bootstrap: [AppComponent]})**

**export class AppModule { }**

**Lesson05 Apply Template in Angular**

**1-we have to know that in the @component decorator we have at least template or template URL**

**2- we can set multiple lines by using this expression ``as following**

**3-we create server component**

**<p>server works!</p>**

**4-we create servers component**

**<p>servers works!</p>**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-servers',**

**//for multiple lines of code we set the expression ``**

**template: `**

**<app-server></app-server>**

**<app-server></app-server>**

**`,**

**styleUrls: ['./servers.component.scss']})**

**export class ServersComponent implements OnInit {**

**constructor() { }**

**ngOnInit() {}}**

**Lesson06 Apply style in Angular**

**1-we can apply inline style on external style file or in template style as below by using [``]**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12">**

**<h3>I'm in the app component</h3><hr>**

**<app-servers></app-servers></div></div></div>**

**//on the code behind we set the following code**

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

**@Component({**

**selector: 'app-root',**

**templateUrl: './app.component.html',**

**//  styleUrls: ['./app.component.scss'],**

**styles:[`**

**h3{color:blue;}**

**`]})**

**export class AppComponent {**

**title = 'my-app';**

**fullName:string = "";}**

**Lesson07 apply custom selector and custom attribute component**

**Notes: -**

**1-we can call component in the following ways :-**

**A-custom class**

**B-custom attribute**

**C-custom selector**

**<!doctype html>**

**<html lang="en">**

**<head>**

**<meta charset="utf-8">**

**<title>MyApp</title>**

**<base href="/">**

**<meta name="viewport" content="width=device-width, initial-scale=1">**

**<link rel="icon" type="image/x-icon" href="favicon.ico">**

**</head>**

**<body>**

**//by using custom selector**

**<app-root></app-root>**

**//by call component using custom class**

**<div class="app-root"></div>**

**//by call component using custom attribute**

**<div app-root></div>**

**</body>**

**</html>**

**On the code behind we set the following code**

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

**@Component({**

**//by using custom selector component**

**selector: 'app-root',**

**//by using custom attribute component**

**selector:'.app-root',**

**//by using custom class component**

**selector:'[app-root]',**

**templateUrl: './app.component.html',**

**styleUrls: ['./app.component.scss']})**

**export class AppComponent {**

**title = 'my-app';**

**fullName:string = "";}**

**Lesson08 what is the Data Binding in Angular 8**

**Notes: -**

**1-Understanding DataBinding**

**DataBinding : is the communication between the typescript and the html (template)**

TypeScript Code

(Business Logic)

**Template**

**(HTML)**

Output Data

**String interpolation {{data}}**

**Property binding [visible]=”data”**

React to (user) Events

**Event Binding (event)=”expression”**

**2 way binding (combine between the output data and react to user events) [(ngModel)]=”data”**

**Lesson09 string Interpolation**

**1-by using the expression {{}} we pass the value from component to html**

**2-we can use function in the interpolation as below**

**//we declare the interpolation it as below**

**<p> Hello {{'hello'}}  {{ fullName }} every body in {{ count }}  --- {{  getStatus() }}  </p>**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-server',**

**templateUrl: './server.component.html',**

**styleUrls: ['./server.component.scss']})**

**export class ServerComponent implements OnInit {**

**//we define the 2 variables one of type string and number of 10000 as below**

**fullName:string = "EveryBody";**

**count:number = 10000;**

**constructor() { }**

**ngOnInit() {}**

**getStatus(){return this.count >= 10000 ? true : false;}}**

**Lesson10 Property Binding in Angular**

**1-string interpolation using with the string while on the property binding using with the any type of data**

**<p> Hello {{'hello'}}  {{ fullName }} every body in {{ count }}  --- {{  getStatus() }}  </p>**

**//we attach the property binding with the variables as below**

**<button [disabled]="isNewUser" class="btn btnprimary" value="Click Me">Click Me</button>**

**On the code behind we set the following code**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-server',**

**templateUrl: './server.component.html',**

**styleUrls: ['./server.component.scss']})**

**export class ServerComponent implements OnInit {**

**isNewUser:boolean = false;**

**fullName:string = "EveryBody";**

**count:number = 10000;**

**constructor() { }**

**ngOnInit() {**

**setTimeout(() => {**

**//after 5 seconds we execute the following code**

**this.isNewUser = true;} , 5000);}**

**getStatus(){return this.count >= 10000 ? true : false;}}**

**Lesson11 Event Binding**

**you can apply event binding from template to typescript as below**

**<button [disabled]="isNewUser" class="btn btn-primary" value="Click Me" (click)="onButtonClick()">Click Me</button>**

**//on the code behind we set the following code**

**onButtonClick(){**

**this.fullName = "You Clicked Me ";}**

**Lesson12 Passing Parameters in the Event Binding**

**//on the html page we set the following code**

**<input type="text" class="form-control" (input)="onTextChange($event)"/>**

**{{textFullName}}**

**//on the code behind we set the following code**

**onTextChange(event:any){**

**this.textFullName = event.target.value;}**

**Lesson13 two-way Binding**

**1-two-way binding is mix mode between the event binding and the property binding**

**2-two-way binding is used to reflect the template and template code behind**

**While the event binding is used to reflect only from template to code behind only**

**3-you must include formModule on the app.moudle as below**

**//we have to import FormsModule on the app.module as below**

**import { FormsModule } from '@angular/forms';**

**@NgModule({**

**declarations: [],**

**imports: [FormsModule],**

**providers: [],**

**bootstrap: [AppComponent]})**

**export class AppModule { }**

**//it will change when you change on textbox 2 or textbox 1**

**<p>with 2 way binding</p>**

**<input type="text"  class="form-control" [(ngModel)]="textFullName" />**

**<br/>**

**//it will change only on the variable on the code behind not the view**

**<p>with 1 way binding</p>**

**<input type="text" class="form-control" (input)="onTextChange($event)"/>**

**{{textFullName}}**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-server',**

**templateUrl: './server.component.html',**

**styleUrls: ['./server.component.scss']})**

**export class ServerComponent implements OnInit {**

**textFullName:string = "initial Full Name";**

**constructor() { }**

**ngOnInit() {}**

**onTextChange(event:any){**

**this.textFullName = event.target.value;}}**

**Lesson14 Understanding Directives**

**Notes: -**

**1-Directives are instructions in the DOM like when we create component and using selector of the component on the html this is called instruction on the DOM instructions that add this component on this html**

**<p appTurnGreen>Hello every body on the whole world</p>**

**@Directive({Selector:’[appTurnGreen]’})**

**Export class TurnGreenDirective{…}**

**Lesson15 using NGIF to Output Data Conditionally**

**1-by using \*ngIf to add / hide DOM element on the html page**

**2-we see that there is hacker point on the DOM element on the browser to keep track where the DOM element must be placed**

**<p>with 2 way binding</p>**

**<input type="text"  class="form-control" [(ngModel)]="fullName" />**

**<br/><p>with 1 way binding</p>**

**<input type="text" class="form-control" (input)="onTextChange($event)"/>**

**<div \*ngIf="status">{{fullName}}</div>**

**<button type="submit" (click)="clickTarget($event)"> Activate</button>**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-server',**

**templateUrl: './server.component.html',**

**styleUrls: ['./server.component.scss']})**

**export class ServerComponent implements OnInit {**

**status:boolean = false;**

**fullName:string = "initial Full Name";**

**constructor() { }**

**ngOnInit() {}**

**onTextChange(event:any){**

**this.fullName = event.target.value;}**

**clickTarget(event){**

**this.status = !this.status;}}**

**Lesson16 Apply ng-template with \*ngIf**

**<div \*ngIf="status;else noServer">**

**<p>The Full Name is {{fullName}}</p>**

**</div>**

**//we use ng-tempalte directive structural to show the content instead of the content of the //div if the condition is false**

**<ng-template #noServer>**

**<p>No Server on this server</p>**

**</ng-template>**

**<button type="submit" (click)="clickTarget($event)"> Activate</button>**

**//on the code behind we set the following code**

**status:boolean = false;**

**clickTarget(event){this.status = !this.status;}**

**Lesson17 Styling Elements Dynamically with ng Style**

**1-the structural directives used to add or remove DOM elements while the attribute directive used to change only the elements where they placed on.**

**2-ngStyle {style property : value} key pairs**

**we apply ngStyle property as following**

**1-on servers.component.html we put the following code**

**<app-server></app-server>**

**<app-server></app-server>**

**2-on server.component.html we put the following code**

**<br/>**

**//we using the method called getColor() on the ng Style property binding**

**<div [ngStyle]="{backgroundColor:getColor()}">{{fullName}} is Unique</div>**

**3-on the code behind server.component.ts we put the following code**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-server',**

**templateUrl: './server.component.html',**

**styleUrls: ['./server.component.scss']})**

**export class ServerComponent implements OnInit {**

**status:boolean = false;**

**constructor(){this.status = Math.random() > 0.5 ? true : false;}**

**ngOnInit() {}**

**//this method return string that represent the color string**

**getColor(){**

**return this.status ? "red" : "green";}}**

**Lesson18 Adding CSS Classes Dynamically**

**1-ngClass {class Name : boolean} key pairs**

**//to show or hide class based on Boolean flag**

**<div  [ngClass]="{online:status}">{{fullName}} is Unique from Ng Class</div>**

**import { Component, OnInit } from '@angular/core';**

**@Component({**

**selector: 'app-server',**

**templateUrl: './server.component.html',**

**styleUrls: ['./server.component.scss']})**

**export class ServerComponent implements OnInit {**

**status:boolean = false;**

**constructor(){this.status = Math.random() > 0.5 ? true : false;}**

**ngOnInit() {}**

**Lesson19 Outputting Lists with ng For**

**//on the html page we set the following code**

**<button type="submit" class="btn btn-primary" (click)="clickMe()">Click Me</button>**

**<div \*ngFor="let server of servers">**

**<app-server></app-server>**

**</div>**

**//on the code behind we set the following code**

**servers:any[] = ["l1","l2"];**

**clickMe(){this.servers.push("l3");}**

**Lesson20 using Index with ng For**

**1-we are using index to get the index number of the item inside the list to apply the style**

**<button type="submit" class="btn btn-primary" (click)="clickMe()">Click Me</button>**

**<div \*ngFor="let server of servers;let i = index" [ngClass]="{'white-text': i >= 4}">**

**[ngStyle]="{backgroundColor: i >= 4 ? 'blue’: 'transparent'}"**

**{{server}}**

**</div>**

**clickMe(){this.servers.push(new Date());}**