**Section02 Basics in Angular**

**Lesson01 How Angular App Loaded and Started**

**Notes:-**

**1-angular is SPA (single page application) which use index.html only and using component navigation to navigate between components**

**2-by default there are root component that called by index.html by using selector**

**3-when apply ng serve it will automatically create bundle javascript section which rebuild all the typescript code on the bottom on the html browser page.**

**Steps:-**

**A-in the first part execute is called main.ts which contains this code**

**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 bootstrap AppModule on the application layer**

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

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

**//2-in the app.module.ts > bootstrap section we verify that AppComponent is the component that we initialized the app which is called on the index.html as selector**

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

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

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

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

**@NgModule({**

**declarations: [**

**AppComponent],**

**imports: [**

**BrowserModule,**

**FormsModule],**

**providers: [],**

**//then**

**bootstrap: [AppComponent]**

**})**

**export class AppModule { }**

**Lesson02 Components are important**

**Notes:-**

**1-Angular in the end is a JavaScript framework changing your DOM (HTML) at runtime**

**2-components is key feature from angular that can used multiple times on the app**

**(Has its own html template and CSS file)**

**3-all the components are added on the app.comonent because its root component and called from index.html**

**Command:-**

**Cd src/app/**

**Mkdir components**

**Cd components**

**Ng g c server**

**4-component parts:-**

**A-Decorator: is features that enhance and specify the type such as component, services, and to configure the selector, html template path, CSS path**

**5-App.Moduel parts**

**1-Decleration: used to register component, pipe, directives**

**2-Imports: used to register modules such as FormsModule,BrowserModule,HttpModule , (can split our application into multiple modules)**

**3-Providers: used to register services**

**4-Bootstrap: used to register entry component (root component)**

**Lesson03 using Custom Component**

**Notes:-**

**1-we see that in the angular.json we see the prefix attribute**

**"prefix": "app",**

**2-we will create two component server , servers and using server inside servers as below**

**<p>servers works!</p>**

**//we can call component multiple times on the other component**

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

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

**Lesson04 working with component template**

**Notes:-**

**1-there are two types of template**

**Template Uri: which redirect to the html path**

**Template: which write inline template**

**2-on the template all the style and html are write on the template whereas template URL splitted the styles from the html page**

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

**@Component({**

**selector: 'app-servers',**

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

**template:`**

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

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

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

**`,**

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

**export class ServersComponent implements OnInit {**

**constructor() { }**

**ngOnInit(): void {}}**

**Lesson05 Working with CSS styles in Angular**

**Notes:-**

**1-there are two way to implement styles on component decorator**

**External style file**

**Inline style**

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

**@Component({**

**selector: 'app-root',**

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

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

**styles:[`**

**h3{color:darkblue}**

**`]**

**})**

**export class AppComponent {**

**title = 'my-app';**

**name = "Ali Ahmad";}**

**Lesson06 understanding the component selector**

**Notes:-**

**1-each selector must be unique and can written in two ways**

**A-CSS selector: this type is limited used per div**

**B-attribute selector: this type is limited used per div**

**C-Suds selector: this type is unlimited used**

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

**@Component({**

**//this is attribute selector**

**selector: '[app-server]',**

**//this is class selector**

**selector: '.app-server',**

**//this is selector element**

**selector: 'app-server',**

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

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

**export class ServerComponent implements OnInit {**

**constructor() { }**

**ngOnInit(): void {}}**

**<p>servers works!</p>**

**//attribute selector called**

**<div app-server></div>**

**<div app-server></div>**

**//class selector called**

**<div class="app-server"></div>**

**<div class="app-server"></div>**

**//selector element called**

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

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

**Lesson07 what is Data Binding?**

**Notes:-**

**1-Databinding is communication between typescript code and template html**

**Types:-**

**A-string interpolation {{data}}: from typescript code to template**

**B-property binding [property] =”data”: from template to typescript code**

**String Interpolation:-**

**Which are used to pass from component to the view**

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

**@Component({**

**selector: 'app-server',**

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

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

**export class ServerComponent implements OnInit {**

**constructor() { }**

**ngOnInit(): void {}**

**serverId:number = 10;**

**name:string = "server name";**

**//we can call function on the interpolation**

**getServerStatus(statusName){**

**return `The Server Status : ${statusName}`;}}**

**on the html template we apply the following code as below**

**<p>server works!</p>**

**<span>{{serverId}}   {{name}}</span>**

**<h3>{{getServerStatus("pending")}}</h3>**

**Lesson08 Property Binding**

**Notes:-**

**1-with property binding we can apply passing from component to view through property as below**

**<button class="btn btn-primary" [disabled]="!allowNewServer">Add Server</button>**

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

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

**On the component**

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

**@Component({**

**selector: 'app-servers',**

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

**styleUrls: ['./servers.component.css']**

**})**

**export class ServersComponent implements OnInit {**

**allowNewServer:boolean = false;**

**constructor() {**

**setTimeout(() => {**

**this.allowNewServer = true;**

**},2000);}**

**ngOnInit(): void {}}**

**2-if you want to view data in the view use interpolation, while if you want to interact with html element property use property binding**

**3-interpolation not working with expression, while in property binding working**

**Lesson09 Event Binding**

**Notes:-**

**1-with event binding the communication happen from the view to the component**

**//we apply event binding which effect on the serverCreationStatus**

**<button class="btn btnprimary" [disabled]="!allowNewServer"**

**(click)="onCreateServer()">Add Server</button>**

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

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

**<h3>{{serverCreationStatus}}</h3>**

**export class ServersComponent implements OnInit {**

**allowNewServer:boolean = false;**

**serverCreationStatus:string = "server before create";**

**constructor() {**

**setTimeout(() => {**

**this.allowNewServer = true;},2000);}**

**ngOnInit(): void {}**

**onCreateServer():void{**

**this.serverCreationStatus = "server created";}}**

**Lesson10 Passing and using data with event binding**

**Notes:-**

**1-we can pass event in order to access to the input value as below**

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

**<p>{{serverName}}</p>**

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

**serverName:string = "";**

**onUpdateServerName(event){**

**console.log(event);**

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

**Lesson11 Important Forms Module is Required for two-way binding**

**Notes:-**

**1-Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel directive. This is done by adding the FormsModule to the imports[] array in the AppModule.**

**2-You then also need to add the import from @angular/forms in the app.module.ts file:**

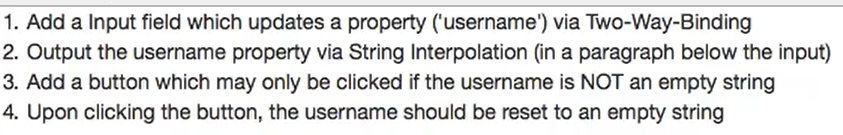
**import { FormsModule } from '@angular/forms';**

**//on the view we apply two-way binding on the property serverName as below**

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

**<p>{{serverName}}</p>**

**Assigment Practicing DataBinding;**



**<p>assigment01 works!</p>**

**<input [(ngModel)]="userName"/>**

**<h2>{{userName}}</h2>**

**<button [disabled]="userName == ''" (click)="resetUserName()">Reset Button</button>**

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

**@Component({**

**selector: 'app-assigment01',**

**templateUrl: './assigment01.component.html',**

**styleUrls: ['./assigment01.component.css']})**

**export class Assigment01Component implements OnInit {**

**userName:string = "";**

**constructor() { }**

**ngOnInit(): void {}**

**resetUserName(){this.userName = "";}}**

**Lesson12 Understanding Directives**

**Notes:-**

**1-directives are instruction in the DOM,once we place this directive selector on html DOM element it will call on that element once reach to it**

**2-directives types**

**A-directive with template (ngIf,ngFor,ngModel)**

**B-directive without template (custom template)**

**<p appTurnGreen>Receives a green background</p>**

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

**Export class TurnGreenDirective{}**

**Lesson13 working with ngIf directive**

**Notes:-**

**1-with ngIf it will remove and add the DOM element**

**<button class="btn btnprimary" [disabled]="!allowNewServer"**

**(click)="onCreateServer()">Add Server</button>**

**<h3 \*ngIf="serverCreated">{{serverCreationStatus}}</h3>**

**export class ServersComponent implements OnInit {**

**allowNewServer:boolean = false;**

**serverCreationStatus:string = "server before create";**

**serverName:string = "";**

**serverCreated:boolean = false;**

**onCreateServer():void{**

**this.serverCreated = true;**

**this.serverCreationStatus = "server created "+this.serverName;}}**

**Lesson14 enhance ngIf with else condition**

**Notes:-**

**1-we can apply ng-template that we can call it on else of ngIf as below**

**<h3 \*ngIf="serverCreated;else noServer">{{serverCreationStatus}}</h3>**

**<ng-template #noServer>**

**<p>No server was created</p>**

**</ng-template>**

**Lesson15 Styling Elements dynamically with ngStyle**

**Notes:-**

**1-there are two types of directives**

**A-Structural directive: like ngIf, ngFor**

**B-attribute directive: like custom directive , ngStyle , ngClass , etc….**

**2-we have to know that [ngStyle] [ngClass] is attribute directive covered by property binding**

**Example:-**

**<p [ngStyle]="{backgroundColor: getColor()}">{{'server'}} with ID {{serverId}} is {{getServerStatus(serverStatus)}}</p>**

**export class ServerComponent implements OnInit {**

**ngOnInit(): void {}**

**constructor() {**

**this.serverStatus = Math.random() > 0.5 ? 'online' : 'offline';}**

**serverId:number = 10;**

**name:string = "server name";**

**serverStatus:string = "offline";**

**getServerStatus(statusName){**

**return `The Server Status : ${statusName}`;}**

**getColor(){**

**return this.serverStatus === 'online' ? 'green' : 'red';}}**

**Lesson16 Applying CSS Classes Dynamically with ngClass**

**Notes:-**

**1-with ngClass its allow to change the class itself**

**<p [ngClass]="{online:serverStatus}"**

**[ngStyle]="{backgroundColor: getColor()}">{{'server'}} with ID {{serverId}} is {{getServerStatus(serverStatus)}}</p>**

**export class ServerComponent implements OnInit {**

**ngOnInit(): void {}**

**constructor() {**

**this.serverStatus = Math.random() > 0.5 ? 'online' : 'offline';}**

**serverId:number = 10;**

**name:string = "server name";**

**serverStatus:string = "offline";**

**getServerStatus(statusName){**

**return `The Server Status : ${statusName}`;}**

**getColor(){**

**return this.serverStatus === 'online' ? 'green' : 'red';}}**

**Lesson17 Output List with NgFor**

**Notes:-**

**1-we can make looping through NgFor as below**

**<app-server \*ngFor="let server of serverLst"></app-server>**

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

**serverLst = ["Server01","Server02"];**

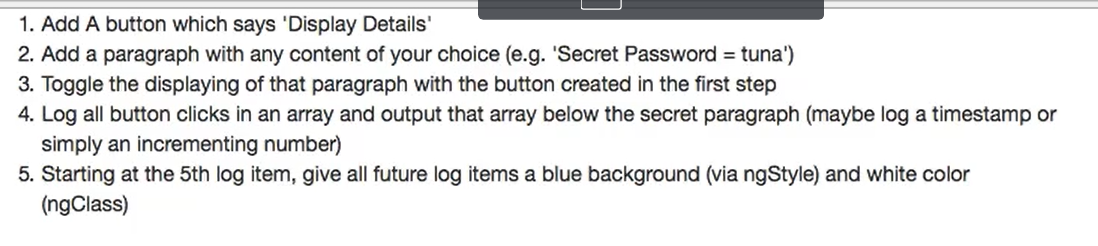
**onCreateServer():void{**

**this.serverCreated = true;**

**this.serverCreationStatus = "server created "+this.serverName;**

**this.serverLst.push(this.serverName);}**

**Assignment 02**



**<p>assigment2 works!</p>**

**<button (click)="showDetails()">Display Details</button>**

**<p \*ngIf="showScret">Secret Password = tuna</p>**

**<div \*ngFor="let log of logs;let i=index"**

**[ngStyle]="{backgroundColor: log > 5 ? 'blue': 'transparent'}">**

**{{log}}</div>**

**import { ThrowStmt } from '@angular/compiler';**

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

**@Component({**

**selector: 'app-assigment2',**

**templateUrl: './assigment2.component.html',**

**styleUrls: ['./assigment2.component.css']})**

**export class Assigment2Component implements OnInit {**

**showScret:boolean = false;**

**logs = [];**

**constructor() { }**

**count:number = 0;**

**ngOnInit(): void {}**

**showDetails(){**

**this.logs.push(new Date());}}**