**Lesson01 Components two-way binding**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12">**

**<p>Add new Servers or blueprints!</p>**

**//define 2 control textbox of apply 2 way binding with assign each control with variable**

**<label>Server Name</label>**

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

**<label>Server Content</label>**

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

**<br>**

**<button class="btn btn-primary" (click)="onAddServer()">Add Server</button>**

**<button class="btn btn-primary" (click)="onAddBlueprint()">Add Server Blueprint</button>**

**</div></div>**

**<hr>**

**<div class="row">**

**<div class="col-xs-12">**

**<div class="panel panel-default" \*ngFor="let element of serverElements">**

**<div class="panel-heading">{{ element.name }}</div>**

**<div class="panel-body">**

**<p>**

**//we make check if the type is server then it will show this DOM element else show another**

**<strong \*ngIf="element.type === 'server'" style="color: red">{{ element.content }}</strong>**

**<em \*ngIf="element.type === 'blueprint'" style="color: blue">{{ element.content }}</em>**

**</p>**

**</div></div></div></div></div>**

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

**export class ServerElementComponent implements OnInit {**

**constructor() { }**

**ngOnInit() {}**

**serverElements = [];**

**newServerName = '';**

**newServerContent = '';**

**onAddServer() {**

**this.serverElements.push({**

**type: 'server',**

**name: this.newServerName,**

**content: this.newServerContent});}**

**onAddBlueprint() {**

**this.serverElements.push({**

**type: 'blueprint',**

**name: this.newServerName,**

**content: this.newServerContent});}}**

**Lesson02 Splitting component in Angular**

**1-we are create the following 2 component as below**

**A-Server-List.component**

**B-Server-entry.component**

**c-app.component**

**(we are using @Input to pass value from parent to child component)**

**(while we are using @Output to pass value from child to parent component)**

**1-on the app.component.html we set the following code**

**<div class="container">**

**//we define the output variable to pass the value from child to parent component**

**<app-server-entry (onSaveUpdate)="onSaveUpdate($event)"></app-server-entry>**

**//this below line represent the list of elements**

**<div \*ngFor="let element of serverElements">**

**//we define the input variable the used to pass the value from parent to child component**

**<app-server-list [element]="element"></app-server-list>**

**</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']})**

**export class AppComponent {**

**//we define this variable that represent the list of element**

**serverElements:any[] = [];**

**onSaveUpdate(event:any){**

**this.serverElements.push(event);}}**

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

**<div class="row">**

**<div class="col-xs-12">**

**<div class="panel panel-default">**

**<div class="panel-heading">{{ element.name }}</div>**

**<div class="panel-body">**

**//we make check if the element type is server or bluePrint type**

**<p><strong \*ngIf="element.type === 'server'" style="color: red">{{ element.content }}</strong>**

**<em \*ngIf="element.type === 'bluePrint'" style="color: blue">{{ element.content }}</em></p>**

**</div></div></div></div>**

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

**export class ServerListComponent implements OnInit {**

**//we define the input variable to pass from child to parent component**

**@Input() element:any;**

**constructor() { }**

**ngOnInit() {}}**

**3-on the server-entry.component.html we set the following code**

**<div class="row">**

**<div class="col-xs-12">**

**<p>Add new Servers or blueprints!</p>**

**<label>Server Name</label>**

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

**<label>Server Content</label>**

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

**<br>**

**<button**

**class="btn btn-primary"**

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

**<button**

**class="btn btn-primary"**

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

**</div>**

**</div>**

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

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

**import { ServerModel } from '../../models/models';**

**@Component({**

**selector: 'app-server-entry',**

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

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

**export class ServerEntryComponent implements OnInit {**

**newServerName:string = "";**

**newServerContent:string = "";**

**//we define the output variable that used to pass value form the child component to //parent component**

**@Output() onSaveUpdate = new EventEmitter();**

**constructor() { }**

**ngOnInit() {}**

**onAddServer(){**

**let obj: any ={**

**type:'server',**

**name:this.newServerName,**

**content:this.newServerContent}**

**this.onSaveUpdate.emit(obj);}**

**onAddBlueprint(){**

**let obj: any ={**

**type:'bluePrint',**

**name:this.newServerName,**

**content:this.newServerContent}**

**this.onSaveUpdate.emit(obj);}}**

**Lesson03 Property Binding in Angular**

**To pass value from html to code behind we have the following 3 ways:-**

**1-HTML elements (Native properties and events)**

**2-Directives (Custom properties and events)**

**3-Components (Custom properties and events)**

**Lesson04 Binding to custom Property**

**To pass value from parent component to child component we using the @Input decorator which represent pass the value from parent component to child component**

**1-on the ServersComponent we set the following code**

**<div class="container">**

**<app-server-entry (onSaveUpdate)="onSaveUpdate($event)"></app-server-entry>**

**<div \*ngFor="let element of serverElements">**

**<app-server-list [element]="element"></app-server-list>**

**</div></div>**

**export class ServersComponent implements OnInit {**

**constructor() { }**

**ngOnInit() {}**

**fullName:string = "";**

**serverElements:any[] = [];**

**onSaveUpdate(event:any){**

**this.serverElements.push(event);}}**

**2-on the ServerEntryComponent we set the following code**

**<div class="row">**

**<div class="col-xs-12">**

**<div class="panel panel-default">**

**<div class="panel-heading">{{ element.name }}</div>**

**<div class="panel-body">**

**<p><strong \*ngIf="element.type === 'server'" style="color: red">{{ element.content }}</strong>**

**<em \*ngIf="element.type === 'bluePrint'" style="color: blue">{{ element.content }}</em>**

**</p></div></div></div></div>**

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

**@Component({**

**selector: 'app-server-list',**

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

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

**export class ServerListComponent implements OnInit {**

**//we define @Input variable which represent custom binding to pass value from parent to //child comoennt**

**@Input() element:any;**

**constructor() { }**

**ngOnInit() {}}**

**Lesson05 Assigning to Alias to custom Property Binding**

**By using the following expression**

**1-On the ServerEntryComponent.ts (child component) we update the below code as**

**//we declare Input variable with alias name as below**

**@Input('aliasElement') element:any;**

**2-On the ServerComponent.html (parent component) we update the following code as**

**<div class="container">**

**<app-server-entry (onSaveUpdate)="onSaveUpdate($event)"></app-server-entry>**

**<div \*ngFor="let element of serverElements">**

**//we see that we are using alias name that used on the child component**

**<app-server-list [aliasElement]="element"></app-server-list>**

**</div></div>**

**Lesson06 Binding to Custom Events**

**1-we are using the @Output to pass value from child component to parent component as below**

**On the component called server-entry.component.html we set the following code**

**<div class="row">**

**<div class="col-xs-12">**

**<p>Add new Servers or blueprints!</p>**

**<label>Server Name</label>**

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

**<label>Server Content</label>**

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

**<br>**

**<button**

**class="btn btn-primary"**

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

**<button**

**class="btn btn-primary"**

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

**</div>**

**</div>**

**//on the code behind we set the event emitter that pass specific type**

**@Output() onSaveUpdate = new EventEmitter<ServerModel>();**

**constructor() { }**

**ngOnInit() {}**

**onAddServer(){**

**let obj: ServerModel = new ServerModel('server',this.newServerName,this.newServerContent);**

**this.onSaveUpdate.emit(obj);}**

**onAddBlueprint(){**

**let obj: ServerModel = new ServerModel('bluePrint',this.newServerName,this.newServerContent);**

**this.onSaveUpdate.emit(obj); }**

**3-on the server.component.html we set the following code**

**<div class="container">**

**//we are assign the output variable with onSaveUpdate method as below**

**<app-server-entry (onSaveUpdate)="onSaveUpdate($event)"></app-server-entry>**

**<div \*ngFor="let element of serverElements">**

**<app-server-list [aliasElement]="element"></app-server-list>**

**</div></div>**

**4-on the server.component.ts we set the following code**

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

**@Component({**

**selector: 'app-servers',**

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

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

**export class ServersComponent implements OnInit {**

**constructor() { }**

**ngOnInit() {}**

**fullName:string = "";**

**serverElements:any[] = [];**

**//on the below code we pass the value from child to parent component as below**

**onSaveUpdate(event:any){**

**this.serverElements.push(event);}}**

**Lesson07 Define Alias Name for Output Variable**

**Steps:-**

**1-on the parent Component we change the output variable name as below**

**<div class="container">**

**<app-server-entry (SaveAlias)="onSaveUpdate($event)"></app-server-entry>**

**<div \*ngFor="let element of serverElements">**

**<app-server-list [aliasElement]="element"></app-server-list>**

**</div></div>**

**2-on the child component we define alias name as below**

**//we define @Ouptut variable with alias name as below  
@Output('SaveAlias') onSaveUpdate = new EventEmitter<ServerModel>();**

**Lesson08 Understanding the view encapsulation**

**View Encapsulation in the Angular means that every .SCSS file the whole classes applied to the component belong only.**

**1-On the child component called server-list.component.html**

**<h6 class="server" \*ngIf="element.type === 'server'">{{ element.content }}</h6>**

**<h6 class="bluePrint" \*ngIf="element.type === 'bluePrint'">{{ element.content }}</h6>**

**//on the CSS file we define the following class as below**

**.server{color: red;}**

**.bluePrint{color:blue;}**

**2-On the child component called server-entry.html**

**<p>Add new Servers or blueprints!</p>**

**//on the CSS file we define the following class as below**

**p{color: green;}**

**3-we see that when open developer tools we see that the 2 classes have the same attribute while the second class have the different attribute**

**//on the first child we see class have it’s own attribute**

**p[\_ngcontent-rcf-c2] { color: green;}**

**//on the second child we see class have it’s own attribute**

**.server[\_ngcontent-rcf-c3] {color: red;}**

**.bluePrint[\_ngcontent-rcf-c3] {color: blue;}**

**Lesson09 More Details in View Encapsulation**

**1-View Encapsulation restrict the class apply to be applied on the component refer to by default**

**None: it will disable Encapsulation by override the CSS class from parent to child with remove the attribute selector on DOM element**

**Native: it will enable encapsulation on specific browser (not preferred)**

**Emulated: it will enable encapsulation on the whole browser (preferred)**

**2-You can change the Encapsulation by apply the CSS classes on the component refer to and its child**

**(this means that when you declare the class on the parent component and you want to make the child component to take the classes from the parent component you can use the encapsulation: None)**

**//on the CSS class file we define the following CSS class as below**

**label{color: red;}**

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

**@Component({**

**selector: 'app-servers',**

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

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

**//we apply the encapsulation none which means that this class apply to the child //component if the class not declared on the own CSS class otherwise it will take the CSS //class from the local CSS and if the encapsulation is Emulated then it will apply the //unique attribute on class to be global on this component only**

**encapsulation:ViewEncapsulation.None})**

**export class ServersComponent implements OnInit {}**

**3-on the child component we define the following class**

**p{color: green;}**

**label{color: yellow;}**

**//the default encapsulation value Is Emulated, the other options is the None , Native**

**@Component({**

**selector: 'app-server-entry',**

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

**styleUrls: ['./server-entry.component.scss'],**

**//encapsulation:ViewEncapsulation.None})**

**export class ServerEntryComponent implements OnInit {}**

**//we see in the end that the child component does not have the attribute CSS selector if the class declared only on the parent otherwise it will define attribute selector with apply the CSS class**

**on the end the encapsulation rule talks about the following**

**A-on the parent component if we make encapsulation : None and set class on the parent component that does not exist on the child component then the child component remove the attribute encapsulation on the DOM element and use the class on the parent component otherwise it will use the local CSS class and apply the encapsulation attribute on the DOM element**

**B-on the parent component if we make encapsulation: Emulated then every component see only the local class and apply the component refer to**

**Lesson10 using Reference variable in Angular**

**Notes: -**

**1-we using reference variable to references only on the template html page**

**Example: -**

**In the below example we define reference variable and pass it as parameter in the code behind**

**<div class="row">**

**<div class="col-xs-12">**

**<p>Add new Servers or blueprints!</p>**

**<label>Server Name</label>**

**//we declare reference variable as below**

**<input type="text" class="form-control" #serverNameEntry>**

**<label>Server Content</label>**

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

**<br>**

**<button**

**class="btn btn-primary"**

**//we pass the reference variable as parameter**

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

**<button**

**class="btn btn-primary"**

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

**</div>**

**</div>**

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

**onAddServer(ele:HTMLInputElement){**

**//on the below command we can access**

**let obj: ServerModel = new ServerModel('server',ele.value,this.newServerContent);**

**this.onSaveUpdate.emit(obj);}**

**Lesson11 using View Child with Reference variable**

**Notes: -**

**1-we can using ViewChild that store the reference variable that refer to DOM element to access to it’s property from code behind as below**

**<div class="row">**

**<div class="col-xs-12">**

**<p>Add new Servers or blueprints!</p>**

**<label>Server Name</label>**

**//we define 2 references on the 2 textboxes as below**

**<input type="text" class="form-control" #serverNameEntry>**

**<label>Server Content</label>**

**<input type="text" class="form-control" #serverContentNameEntry><br>**

**<button class="btn btn-primary"**

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

**<button class="btn btn-primary"**

**(click)="onAddBlueprint(serverNameEntry)">Add Server Blueprint</button></div></div>**

**import { Component, OnInit, Output, ViewEncapsulation, ViewChild, ElementRef } from '@angular/core';**

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

**import { ServerModel } from '../../models/models';**

**@Component({**

**selector: 'app-server-entry',**

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

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

**export class ServerEntryComponent implements OnInit {**

**newServerName:string = "";**

**newServerContent:string = "";**

**@Output('SaveAlias') onSaveUpdate = new EventEmitter<ServerModel>();**

**@ViewChild('serverContentNameEntry', {static: false}) contentDOM: ElementRef;**

**constructor() { }**

**ngOnInit() {}**

**onAddServer(ele:HTMLInputElement){**

**let obj: ServerModel = new ServerModel(**

**'server',ele.value,this.contentDOM.nativeElement.value,);**

**this.onSaveUpdate.emit(obj);}**

**onAddBlueprint(ele:HTMLInputElement){**

**let obj: ServerModel = new ServerModel(**

**'bluePrint',**

**//we can access to the DOM element one by passed as parameter and another is by the //view child property**

**ele.value,**

**this.contentDOM.nativeElement.value);**

**this.onSaveUpdate.emit(obj);}}**

**Lesson12**