**Section05 Understanding Directive**

**Notes:-**

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

**A-attribute directive: which the directive is set as attribute on the html DOM element  
B-structural directive: which effect on the whole DOM element such as ng If , ng For , etc..**

|  |  |
| --- | --- |
| **Attribute directives** | **Structural directives** |
| **Look like normal html attribute (possibly with databinding or event binding)** | **Look like normal html attribute but have leading \* (for desugaring)** |
| **Only effect change the element that are added to** | **Affect whole area in the DOM elements get added / removed** |

**Lesson01 ngFor and ngIf Recap**

**Notes:-**

**1-ngFor and ngIf is structural directives which means ngFor is created DOM element and ngIF show or hide elements**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12">**

**<button class="btn btn-primary" (click)="onlyOdd = !onlyOdd">Only show odd numbers</button>**

**<br><br>**

**<ul class="list-group">**

**<div \*ngIf="onlyOdd">**

**<li class="list-group-item" \*ngFor="let odd of oddNumbers">{{odd}}</li>**

**</div>**

**<div \*ngIf="!onlyOdd">**

**<li class="list-group-item" \*ngFor="let even of evenNumbers">{{even}}</li>**

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

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

**@Component({**

**selector: 'app-root',**

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

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

**export class AppComponent {**

**oddNumbers = [1,3,5,7,9];**

**evenNumbers = [2,4,6,8,10];**

**onlyOdd = false;}**

**Lesson02 ngClass and ngStyle**

**Notes:-**

**1-ngClass and ngStyle are attribute directives which means that it will effect on the DOM properties not itself**

**<ul class="list-group">**

**<li class="list-group-item" \*ngFor="let no of numbers"**

**[ngClass]="{odd:no % 2 !== 0}"**

**[ngStyle]="{backgroundColor: no %2 !== 0 ? 'blue' : 'transparent'}">**

**{{no}}</li></ul>**

**On code behind**

**numbers = [1,2,3,4,5,6,7,8,9,10];**

**Lesson03 Creating basic attribute directive**

**Notes:-**

**1-we want to build custom directive which highlight element**

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

**@Directive({**

**selector: '[appBasicHighlightDirective]'})**

**export class BasicHighlightDirectiveDirective implements OnInit{**

**//we make private automatic property**

**//we inject elementRef as DI on the constructor on the custom directive**

**constructor(private elementRef:ElementRef){ }**

**ngOnInit(): void {this.elementRef.nativeElement.style.backgroundColor = "green";}}**

**//on the view we set the directive selector as below**

**<p appBasicHighlightDirective>Style me with basic directive!</p>**

**//we inject the custom directive on the app.module.ts**

**@NgModule({**

**declarations: [**

**AppComponent,**

**BasicHighlightDirectiveDirective],**

**Lesson04 using the Renderer to build better attribute directive**

**Notes:-**

**1-angular provide better way to access the DOM element with using Renderer**

**(Because the direct way of access DOM element may cause error, while with using Renderer provide secure way to change style of the DOM element)**

**import { Directive, ElementRef, OnInit, Renderer2 } from '@angular/core';**

**@Directive({selector: '[appBetterHighlight]'})**

**export class BetterHighlightDirective implements OnInit {**

**constructor(private renderer:Renderer2,private eleRef:ElementRef) { }**

**//with the renderer injected on the custom directive we give access to make styling on the DOM**

**ngOnInit(): void {**

**this.renderer.setStyle(this.eleRef.nativeElement,"background-color","blue");}}**

**declarations: [**

**AppComponent,**

**BetterHighlightDirective],**

**<p appBetterHighlight>Style me with better directive!</p>**

**Lesson05 using the host listener to listen host event**

**Notes:-**

**1-with @HostListener you can provide listener to the events like mouse enter , mouse over , mouse click ,etc..**

**import { Directive, ElementRef, HostListener, OnInit, Renderer2 } from '@angular/core';**

**@Directive({selector: '[appBetterHighlight]'})**

**export class BetterHighlightDirective implements OnInit {**

**constructor(private renderer:Renderer2,private eleRef:ElementRef) { }**

**ngOnInit(): void {this.renderer.setStyle(this.eleRef.nativeElement,"background-color","blue");}**

**//with host listener we give access to the element with assign to events as below**

**@HostListener('mouseenter') mouseover(eventData:Event){**

**this.renderer.setStyle(this.eleRef.nativeElement,"background-color","green");}**

**@HostListener('mouseleave') mouseleave(eventData:Event){**

**this.renderer.setStyle(this.eleRef.nativeElement,"background-color","red");}}**

**Lesson06 using Host Binding to bind to host properties**

**Notes:-**

**1-there is better way than Renderer called @HostBiniding which provide better way to change style of DOM element**

**import { Directive, ElementRef, HostBinding, HostListener, OnInit, Renderer2 }**

**from '@angular/core';**

**@Directive({selector: '[appBetterHighlight]'})**

**export class BetterHighlightDirective implements OnInit {**

**constructor(private renderer:Renderer2,private eleRef:ElementRef) { }**

**//we set which style we want to apply @HostBinding**

**@HostBinding('style.backgroundColor') backgroundColor:string;**

**ngOnInit(): void {this.renderer.setStyle(this.eleRef.nativeElement,"background-color","blue");}**

**@HostListener('mouseenter') mouseover(eventData:Event){**

**// this.renderer.setStyle(this.eleRef.nativeElement,"background-color","green");**

**this.backgroundColor = 'green';}**

**@HostListener('mouseleave') mouseleave(eventData:Event){**

**// this.renderer.setStyle(this.eleRef.nativeElement,"background-color","red");**

**this.backgroundColor = 'red';}}**

**Lesson07 Binding to directive properties**

**Notes:-**

**1-we can pass input parameter on the custom directive as below**

**<p appPropHighlight [defaultColor]="'gray'" [highlightColor]="'yellow'">Style me with prop directive!</p>**

**import { Directive, ElementRef, HostBinding, HostListener, Input, Renderer2 }**

**from '@angular/core';**

**@Directive({selector: '[appPropHighlight]'})**

**export class PropHighlightDirective {**

**constructor() { }**

**@Input() defaultColor:string = "transparent";**

**@Input() highlightColor:string = "blue";**

**//we set which style we want to apply @HostBinding**

**@HostBinding('style.backgroundColor') backgroundColor:string;**

**ngOnInit(): void {**

**this.backgroundColor = this.defaultColor;}**

**@HostListener('mouseenter') mouseover(eventData:Event){**

**this.backgroundColor = this.highlightColor;}**

**@HostListener('mouseleave') mouseleave(eventData:Event){**

**this.backgroundColor = this.defaultColor;}}**

**Lesson08 what happened behind scenes on structural directives**

**Notes:-**

**1-\*ngFor the star will be transform equivalent to property binding or event binding**

**Which internally <ng-template> and then create multiple DOM elements based on it**

**//ngif before convert**

**<div \*ngIf="!onlyOdd">**

**<li class="list-group-item" \*ngFor="let even of evenNumbers">{{even}}</li></div>**

**//after convert**

**<ng-template [ngIf]="!onlyOdd">**

**<div><li class="list-group-item" \*ngFor="let even of evenNumbers">{{even}}</li></div>**

**</ng-template>**

**Lesson09 Building Structural Directive**

**Notes:-**

**1-with structural directive we can show or hide elements with using TemplateRef<any> , ViewContainerRef which used to inject container with embedded templates**

**import { Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';**

**@Directive({selector: '[appUnless]'})**

**export class UnlessDirective {**

**//we define Input() property which change the visibility of the DOM element**

**@Input() set appUnless(condition:boolean){**

**if(!condition){**

**//it will create template container that hold multiple embeded templates**

**this.vcRef.createEmbeddedView(this.templateRef);}**

**else{this.vcRef.clear();}}**

**constructor(private templateRef:TemplateRef<any>,private vcRef:ViewContainerRef) { }}**

**//on the view with using \*appUnless**

**<div \*appUnless="!onlyOdd">**

**<li class="list-group-item" \*ngFor="let even of evenNumbers">{{even}}</li>**

**</div>**

**Lesson10 understanding ngSwitch**

**Notes:-**

**1-ngSwitch is also structural directive with the benefit of using of multiple check of values as below**

**<div [ngSwitch]="value">**

**<p \*ngSwitchCase="5">Value is 5</p>**

**<p \*ngSwitchCase="10">Value is 10</p>**

**<p \*ngSwitchCase="15">Value is 15</p>**

**<p \*ngSwitchCase="20">Value is 20</p>**

**<p \*ngSwitchCase="25">Value is 25</p>**

**<p \*ngSwitchCase="30">Value is 30</p>**

**<p \*ngSwitchDefault>Default is 0</p>**

**</div>**

**value= 10;**