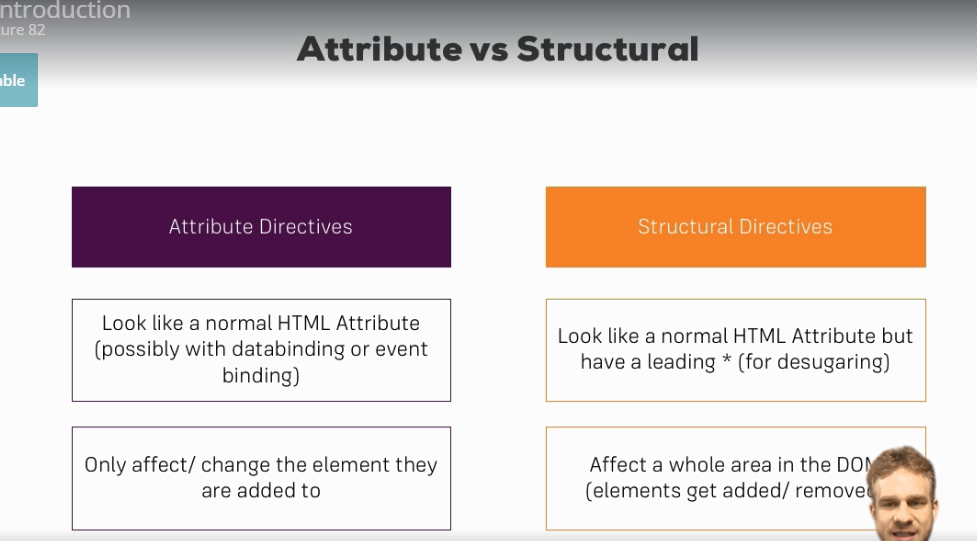
**Section 7: Directives Deep Dive**

**Section 7: Lecture 82 //Module Introduction**

1. Let’s repeat quickly what we have learnt this far. We learnt how to build our own directives. We will really dive into the difference between structural and attribute directives and what this star means on structural directives.
2. Attribute vs Structural Directives. Attribute directives are called like this because they sit on elements just like attributes. Structural directives are also same but they basically change the structure of the DOM around this element.
3. If you have \*ngIf on a paragraph, if this condition is false then this paragraph is removed from the DOM. So, the overall view container is affected.
4. Whereas in the case of attribute directive you never destroy an element from the DOM, you only change properties of that element – for Example the background color. So, we have attribute directives only affecting the element they sit on.
5. Structural directives which affect the whole DOM. The structural directives affect the whole area around the element on which they were placed on. This was the difference between the two types of directives.
6. First we will talk about the directives we know and how to use them. Then we will dive deeper into creating our own directives.



**Section 7: Lecture 83//ngFor and ngIf Recap**

1. We will create a simple project now to toggle the odd numbers.
2. Please add below code for printing odd and even numbers along with toggle button. app.component.html:
3. <div class="container">
4. <div class="row">
5. <div class="col-xs-12">
6. <button
7. class = "btn btn-primary"
8. (click) = "onlyOdd = !onlyOdd"
9. >Only show odd numbers</button>
10. <br><br>
12. <ul class="list-group">
13. <div \*ngIf="onlyOdd">
14. <li class="list-group-item"
15. \*ngFor="let odd of oddNumbers"
16. >{{ odd }}</li>
17. </div>
18. <div \*ngIf="!onlyOdd">
19. <li class="list-group-item"
20. \*ngFor="let even of evenNumbers"
21. >{{ even }}</li>
22. </div>
24. </ul>
25. </div>
26. </div>
27. </div>

3. app.component.ts:

import {Component} from '@angular/core'

@Component({

selector: 'app-root',

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

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

})

export class AppComponent{

//numbers = [1,2,3,4,5];

oddNumbers = [1,3,5];

evenNumbers = [2,4];

onlyOdd = false;

}

**Section 7: Lecture 84 //ngClass and ngStyle Recap**

1. Now, we will display the use of ngClass and ngStyle using property binding.
2. app.component.html:
3. <div class="container">
4. <div class="row">
5. <div class="col-xs-12">
6. <button
7. class = "btn btn-primary"
8. (click) = "onlyOdd = !onlyOdd"
9. >Only show odd numbers</button>
10. <br><br>
12. <ul class="list-group">
13. <div \*ngIf="onlyOdd">
14. <li class="list-group-item"
15. [ngClass]="{odd: odd % 2 !== 0}"
16. [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
17. \*ngFor="let odd of oddNumbers"
18. >{{ odd }}</li>
19. </div>
20. <div \*ngIf="!onlyOdd">
21. <li class="list-group-item"
22. [ngClass]="{odd: even % 2 !== 0}"
23. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
24. \*ngFor="let even of evenNumbers"
25. >{{ even }}</li>
26. </div>
28. </ul>
29. </div>
30. </div>
31. </div>

**Section 7: Lecture 85//Creating a Basic Attribute Directive**

1. Here we will create a separate folder for the directive, in the folder we will create a new basic-highlight.directive.ts file.
2. In the directive file we will create a new directive by using @Directive and import it from ‘@angular/core’. This will contain the info about the selector of the directive. We will wrap the selector in the square brackets such that it acts like attribute. We have added the square brackets such that it can be recognized when we use it without square brackets.
3. If we don’t put square brackets then that would be select by element. Once, we put square brackets around the directive selector then it would be recognized whenever I add appBasicHeighlight without square brackets to an element.
4. Now, to prove it we need to give some visual clue that it is working. The most basic use case is that we change the background color of the element where we attach this directive. For this we need the access to the element on which the directive sits on and the cool thing is that angular gives us this access.
5. We can inject the element on which the directive sits on into this directive.
6. And on injection we will look the closer look at in the next course module, which is about services. It is easy way of getting access to some other class without having instantiate it on our own.
7. We do inject by adding the constructor which every class has, we don’t need to write anything in the constructor body for now. But here in the list of arguments we want to enlist the arguments whatever the instance here of the class is created, and of course angular is responsible for creating these instances. Therefore if we tell it to give us a specific type of argument – this is what injection is – angular will try to create this thing we need and give it us.
8. In this case that thing is reference to the element on which the directive was placed on.
9. So, we will use ElementRef: ElementRef ; here type is important
10. Here we can use the typescript feature by making the argument private which in turn make it the feature of this class.
11. Now, with that we got access to the element, now we can use it here in the constructor for example – access the native element and then do something with it. Though better place then constructor is OnInit.
12. Just like the component the directive also has the ngOnOnit lifecycle hook. It doesn’t have all the other hooks though.
13. Directive unlike a component doesn’t have a view and also doesn’t have a template.
14. In the ngOnInit we can access the elementRef by using this.elementRef.nativeElement.
15. Here we are getting access to the element on which the directive was placed on, and then we are overriding the style of that element.
16. Now let’s use this directive - to use this directive we have to do 2 things
17. Like for a component we need to inform angular that we have a new directive, just like components it doesn’t scan all our files so it doesn’t know. So, let’s go to app.module.ts and inside the declarations we have to add out BasicHighlightDirective and also add the import pointing to the basic-highlight folder.
18. Now, since we informed angular about or directive, we can use it in our app. We will add it in app.component.html template.
19. So this was our first basic attribute directive.
20. app.module.ts
21. import { BrowserModule } from '@angular/platform-browser';
22. import { NgModule } from '@angular/core';
23. import { FormsModule } from '@angular/forms';
24. import { AppComponent } from './app.component';
25. import { ServerComponent } from './server/server.component';
26. import { ServersComponent } from './servers/servers.component';
27. import { HeaderComponent } from './header/header.component';
28. import { RecipesComponent } from './recipes/recipes.component';
29. import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';
30. import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
31. import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';
32. import { ShoppingListComponent } from './shopping-list/shopping-list.component';
33. import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
34. import { CockpitComponent } from './cockpit/cockpit.component';
35. import { ServerElementComponent } from './server-element/server-element.component';
36. import { BasicHighlightDirective } from './basic-highlight/basic-highlight.directive';
37. @NgModule({ //Component Decorator
38. declarations: [
39. AppComponent,
40. ServerComponent,
41. ServersComponent,
42. HeaderComponent,
43. RecipesComponent,
44. RecipeListComponent,
45. RecipeDetailComponent,
46. RecipeItemComponent,
47. ShoppingListComponent,
48. ShoppingEditComponent,
49. CockpitComponent,
50. ServerElementComponent,
51. BasicHighlightDirective
52. ],
53. imports: [
54. BrowserModule,
55. FormsModule
56. ],
57. providers: [],
58. bootstrap: [AppComponent]
59. })
60. export class AppModule { }

19. app.component.html:

<!-- Aplication one -->

<!--<input type="text" [(ngModel)]="name">

<p>{{ name }}</p>-->

<!--<div class="container">

<div class="row">

<div class="col-xs-12">

<h3>I am in the AppComponent!!!</h3>-->

<!-- <div app-servers></div>-->

<!--<app-header></app-header>-->

<!--<div class="app-servers"></div>-->

<!-- </div>

</div>

</div>-->

<!-- Application 2 -->

<!--<app-header></app-header>

<div class="container">

<div class="row">

<div class="col-md-12">

<app-recipes></app-recipes>

<app-shopping-list></app-shopping-list>

</div>

</div>

</div>-->

<!-- Application 3 -->

<!-- <div class="container">

<app-cockpit

(serverCreated)="onServerAdded($event)"

(bpCreated)="onBlueprintAdded($event)"

></app-cockpit>

<hr>

<div class="row">

<div class="col-md-12">

<button class="btn btn-primary" (click)="onChangeFirst()">Change First Element</button>

<button class="btn btn-danger" (click)="onDestroyFirst()">Destroy first Component</button>

<app-server-element

\*ngFor="let serverElement of serverElements"

[srvElement]="serverElement"

[name]="serverElement.name"

>

<p #contentParagraph>

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

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

</p>

</app-server-element>

</div>

</div>

</div> -->

<!-- Unit 7 - Directives Deep Dive; application 4 -->

<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"

[ngClass]="{odd: odd % 2 !== 0}"

[ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"

\*ngFor="let odd of oddNumbers"

>{{ odd }}</li>

</div>

<div \*ngIf="!onlyOdd">

<li class="list-group-item"

[ngClass]="{odd: even % 2 !== 0}"

[ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"

\*ngFor="let even of evenNumbers"

>{{ even }}</li>

</div>

</ul>

<p appBasicHighlight>Style me with basic directive!</p>

</div>

</div>

</div>

20. basic-heighlight.directive.ts :

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

@Directive({

selector: '[appBasicHighlight]'

})

export class BasicHighlightDirective implements OnInit {

constructor(private elementRef: ElementRef){

elementRef.nativeElement

}

ngOnInit(){

this.elementRef.nativeElement.style.backgroundColor = 'green';

}

}

**Section 7: Lecture 86//Using the Renderer to build a better Attribute Directive**

1. In the last lecture we triggered out first directive and its doing its Job. We prepared the selector and how to use that selector, we set up our attribute directive and we also learnt how to register it into app module.
2. Now, as we have seen that we have accessed the element of the directive and changed its style to green by assigning the background color directly, however, we had discussed earlier that this kind of practice is not good. We should use a different tool that we will look in a second.
3. Angular is actually able to render your templates without a DOM, in that case these properties might not be available. It could do this when using service workers, some other advanced use cases. But none the less it’s not a good practice to directly access your elements.
4. How should we access them then?; well, there is another helper which you can check and that is the **renderer**, so let’s do this.
5. Let’s not do this in our basic-highlighter, but let’s create a new directive with the generate command.
6. **ng g directive better-highlight or ng g d better-highlight**
7. We can also create central shared or directives folder, which would contain all the directives. Now, for the demo purposes we have created a clear separation which makes it very easy to see as we have put them in the individual folders.
8. Here we are using the renderer for changing the background color of the paragraph; now, why it is better approach?; Angular is not limited to running in the browser it, for example, also works with service workers - these are the environments where you might not have the access to the DOM. So, if we try to change the DOM as we did in the basic highlight by directly accessing the native element and the style of this element you might get an error in some circumstances.
9. Now, to be honest in most circumstances you probably don’t end you probably also know if you app is going to run in the browser or not – still it’s a better practice to use a renderer for accessing the DOM and also use the methods which renderer provides to access the DOM.
10. We can also learn more about the renderer in the article after this lecture here.
11. app.component.html:
12. <div class="container">
13. <div class="row">
14. <div class="col-xs-12">
15. <button
16. class = "btn btn-primary"
17. (click) = "onlyOdd = !onlyOdd"
18. >Only show odd numbers</button>
19. <br><br>
21. <ul class="list-group">
22. <div \*ngIf="onlyOdd">
23. <li class="list-group-item"
24. [ngClass]="{odd: odd % 2 !== 0}"
25. [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
26. \*ngFor="let odd of oddNumbers"
27. >{{ odd }}</li>
28. </div>
29. <div \*ngIf="!onlyOdd">
30. <li class="list-group-item"
31. [ngClass]="{odd: even % 2 !== 0}"
32. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
33. \*ngFor="let even of evenNumbers"
34. >{{ even }}</li>
35. </div>
36. </ul>
37. <p appBasicHighlight>Style me with basic directive!</p>
38. <!-- appBetterHighlight -->
39. <p appBetterHighlight>Style me with a better directive!</p>
40. </div>
41. </div>
42. </div>

12. better-highlight.directive.ts:

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

@Directive({

selector: '[appBetterHighlight]'

})

export class BetterHighlightDirective implements OnInit {

constructor(private elRef:ElementRef,private renderer: Renderer2) { }

ngOnInit(){

this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');

};

}

13. app.module.ts:

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

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

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

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

import { ServerComponent } from './server/server.component';

import { ServersComponent } from './servers/servers.component';

import { HeaderComponent } from './header/header.component';

import { RecipesComponent } from './recipes/recipes.component';

import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';

import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';

import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';

import { CockpitComponent } from './cockpit/cockpit.component';

import { ServerElementComponent } from './server-element/server-element.component';

import { BasicHighlightDirective } from './basic-highlight/basic-highlight.directive';

import { BetterHighlightDirective } from './better-highlight/better-highlight.directive';

@NgModule({ //Component Decorator

declarations: [

AppComponent,

ServerComponent,

ServersComponent,

HeaderComponent,

RecipesComponent,

RecipeListComponent,

RecipeDetailComponent,

RecipeItemComponent,

ShoppingListComponent,

ShoppingEditComponent,

CockpitComponent,

ServerElementComponent,

BasicHighlightDirective,

BetterHighlightDirective

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Section 7: Lecture 87//More about renderer**

1. In the last lecture, we used the Angular Renderer class to change the style of a HTML element. As explained in that lecture, you should use the Renderer for any DOM manipulations.
2. Of course, you can do more than simply change the styling of an element via setStyle(). Learn more about the available Renderer methods [here](https://angular.io/docs/ts/latest/api/core/index/Renderer2-class.html).

**Section 7: Lecture 88//Using HostLiner to Listen to Host Events**

1. Now, as we created 2 directives in the previous lecture, where we have blue and green backgrounds; now we want to change the functionality and make it blue only when we hover over it.
2. So, let’s improve the better highlight directive here. We need to react to some events occurring on the element the directive sits on and the quick and easy way to do this inside of this directive is to simply add a new decorator. So, we will add @HostListener and add it to some method you want to execute; HostListener here takes argument name as the input and that would be mousenter – this is one of the events supported by the DOM element on which this directive sits on. Basically we have all the events available- we can use any of the events and we can also use this with event binding before.
3. We can also receive event data here i.e. in the mouseover method – mouseover(eventData: Event) - it receives eventDataas the argument which is of the type event. We can also listen to custom events here and retrieve that data.
4. That is just like a method that you execute like click listener or whatever your event is and then pass the method between the quotation marks - that is just happening here – HostListener is just a convenient way to listen that events in that method.
5. So, with that we listen to mouse enter event, we get the event data – which I don’t need here, what I want to do here is – I want to change the background color of the element.
6. Better-highlight.directive.ts:
7. import { Directive, Renderer2, OnInit, ElementRef, HostListener } from '@angular/core';
8. @Directive({
9. selector: '[appBetterHighlight]'
10. })
11. export class BetterHighlightDirective implements OnInit {
13. constructor(private elRef:ElementRef,private renderer: Renderer2) { }
15. ngOnInit(){
16. //this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');
17. }
18. @HostListener('mouseenter') mouseover(){
19. this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');
20. }
21. @HostListener('mouseleave') mouseleave(){
22. this.renderer.setStyle(this.elRef.nativeElement, 'background-color','transparent');
23. }
24. }

**Section 7: Lecture 89//Using HostBinding to Bind to Host Properties**

1. There is another decorator which we can use – which lets us not use the renderer.
2. There is nothing wrong in using the renderer, but we have got even easier way to do this i.e. simply changing the background color if this is all that we want to do. The decorator we are talking about is @HostBinding
3. Better-highlight.directive.ts:
4. import { Directive, Renderer2, OnInit, ElementRef, HostListener, HostBinding } from '@angular/core';
5. @Directive({
6. selector: '[appBetterHighlight]'
7. })
8. export class BetterHighlightDirective implements OnInit {
10. @HostBinding('style.backgroundColor') backgroundColor: string = 'transparent';
11. constructor(private elRef:ElementRef,private renderer: Renderer2) { }
13. ngOnInit(){
14. //this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');
15. }
16. @HostListener('mouseenter') mouseover(){
17. //this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');
18. this.backgroundColor = 'blue';
19. }
20. @HostListener('mouseleave') mouseleave(){
21. //this.renderer.setStyle(this.elRef.nativeElement, 'background-color','transparent');
22. this.backgroundColor = 'transparent';
23. }
24. }

**Section 7: Lecture 90//Binding to directive Properties**

1. Our directive is looking really great we are almost there but there is one other thing I want to add i.e. one functionally I want to add. Now, we can see that the background color of that paragraph is now dynamic; however, we cannot decide that which color background we want.
2. There developer should be able to dynamically set the value, maybe we even want to do this in our own app – suppose the color we are setting changes according to some other parameter in our app. Right now, the color is hard codded in there. So, that is something we can improve with the tools we have already learnt i.e. custom property binding, info – custom event binding also works with directives.
3. So, how can we use custom property binding then? Let’s add to properties to which we bind.
4. app.component.html:
5. <!-- Aplication one -->
6. <!--<input type="text" [(ngModel)]="name">
7. <p>{{ name }}</p>-->
8. <!--<div class="container">
9. <div class="row">
10. <div class="col-xs-12">
11. <h3>I am in the AppComponent!!!</h3>-->
13. <!-- <div app-servers></div>-->
14. <!--<app-header></app-header>-->
15. <!--<div class="app-servers"></div>-->
16. <!-- </div>
17. </div>
18. </div>-->
19. <!-- Application 2 -->
20. <!--<app-header></app-header>
21. <div class="container">
22. <div class="row">
23. <div class="col-md-12">
24. <app-recipes></app-recipes>
25. <app-shopping-list></app-shopping-list>
26. </div>
27. </div>
28. </div>-->
29. <!-- Application 3 -->
30. <!-- <div class="container">
31. <app-cockpit
32. (serverCreated)="onServerAdded($event)"
33. (bpCreated)="onBlueprintAdded($event)"
34. ></app-cockpit>
36. <hr>
37. <div class="row">
38. <div class="col-md-12">
39. <button class="btn btn-primary" (click)="onChangeFirst()">Change First Element</button>
40. <button class="btn btn-danger" (click)="onDestroyFirst()">Destroy first Component</button>
41. <app-server-element
42. \*ngFor="let serverElement of serverElements"
43. [srvElement]="serverElement"
44. [name]="serverElement.name"
45. >
46. <p #contentParagraph>
47. <strong \*ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>
48. <em \*ngIf="serverElement.type === 'blueprint'" style="color: blue">{{ serverElement.content }}</em>
49. </p>
50. </app-server-element>
51. </div>
52. </div>
53. </div> -->
54. <!-- Unit 7 - Directives Deep Dive; application 4 -->
55. <div class="container">
56. <div class="row">
57. <div class="col-xs-12">
58. <button
59. class = "btn btn-primary"
60. (click) = "onlyOdd = !onlyOdd"
61. >Only show odd numbers</button>
62. <br><br>
64. <ul class="list-group">
65. <div \*ngIf="onlyOdd">
66. <li class="list-group-item"
67. [ngClass]="{odd: odd % 2 !== 0}"
68. [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
69. \*ngFor="let odd of oddNumbers"
70. >{{ odd }}</li>
71. </div>
72. <div \*ngIf="!onlyOdd">
73. <li class="list-group-item"
74. [ngClass]="{odd: even % 2 !== 0}"
75. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
76. \*ngFor="let even of evenNumbers"
77. >{{ even }}</li>
78. </div>
79. </ul>
80. <p appBasicHighlight>Style me with basic directive!</p>
81. <!-- appBetterHighlight -->
82. <p appBetterHighlight [defaultColor]="'yellow'" [highlightColor]="'red'">Style me with a better directive!</p>
83. </div>
84. </div>
85. </div>

5. better-highlight.directive.ts:

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

@Directive({

selector: '[appBetterHighlight]'

})

export class BetterHighlightDirective implements OnInit {

@Input() defaultColor: string= 'transparent';

@Input() highlightColor: string= 'blue'; //these value can be overrriden from outside

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

constructor(private elRef:ElementRef,private renderer: Renderer2) { }

ngOnInit(){

this.backgroundColor = this.defaultColor;

//this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');

}

@HostListener('mouseenter') mouseover(){

//this.renderer.setStyle(this.elRef.nativeElement, 'background-color','blue');

this.backgroundColor = this.highlightColor;

}

@HostListener('mouseleave') mouseleave(){

//this.renderer.setStyle(this.elRef.nativeElement, 'background-color','transparent');

this.backgroundColor = this.defaultColor;

}

}

**Section7: Lecture 91//What happens behind the scenes on Structural Directives.**

1. Now, we had a closer look at the attribute directives and created our own attribute directives before creating our own structural directives let us understand why the \* is required before the structural directives.
2. The start i.e.\* indicates that this is a structural directive, but the point is why does it needs to know that it is a structural directive, because structural directives with \* are nicer way to use them actually. Behind the scene angular will transform them into something else. Because in all things such as property binding, string interpolation there is no \*. So, it internally converts this into these tools such as property binding.
3. If we write it differently we will end up as below i.e. app.component.html:
4. <div class="container">
5. <div class="row">
6. <div class="col-xs-12">
7. <button
8. class = "btn btn-primary"
9. (click) = "onlyOdd = !onlyOdd"
10. >Only show odd numbers</button>
11. <br><br>
13. <ul class="list-group">
14. <div \*ngIf="onlyOdd">
15. <li class="list-group-item"
16. [ngClass]="{odd: odd % 2 !== 0}"
17. [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
18. \*ngFor="let odd of oddNumbers"
19. >{{ odd }}</li>
20. </div>
21. <div \*ngIf="!onlyOdd">
22. <li class="list-group-item"
23. [ngClass]="{odd: even % 2 !== 0}"
24. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
25. \*ngFor="let even of evenNumbers"
26. >{{ even }}</li>
27. </div>
28. <ng-template [ngIf]="!onlyOdd">
29. <div>
30. <li class="list-group-item"
31. [ngClass]="{odd: even % 2 !== 0}"
32. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
33. \*ngFor="let even of evenNumbers"
34. >{{ even }}</li>
35. </div>
36. </ng-template>
37. </ul>
38. <p appBasicHighlight>Style me with basic directive!</p>
39. <!-- appBetterHighlight -->
40. <p appBetterHighlight [defaultColor]="'yellow'" [highlightColor]="'red'">Style me with a better directive!</p>
41. </div>
42. </div>
43. </div>

**Section 7: Lecture 92 //building a structural Directive**

1. Now, we understand what the role or the star is we can write our own structural directive.
2. Now, let’s create a new directive and we will use the CLI for this. We use below command for it:

ng g d unless

1. Now, in the directive we need to get the condition as the input.
2. app.component.ts:
3. <!-- Aplication one -->
4. <!--<input type="text" [(ngModel)]="name">
5. <p>{{ name }}</p>-->
6. <!--<div class="container">
7. <div class="row">
8. <div class="col-xs-12">
9. <h3>I am in the AppComponent!!!</h3>-->
11. <!-- <div app-servers></div>-->
12. <!--<app-header></app-header>-->
13. <!--<div class="app-servers"></div>-->
14. <!-- </div>
15. </div>
16. </div>-->
17. <!-- Application 2 -->
18. <!--<app-header></app-header>
19. <div class="container">
20. <div class="row">
21. <div class="col-md-12">
22. <app-recipes></app-recipes>
23. <app-shopping-list></app-shopping-list>
24. </div>
25. </div>
26. </div>-->
27. <!-- Application 3 -->
28. <!-- <div class="container">
29. <app-cockpit
30. (serverCreated)="onServerAdded($event)"
31. (bpCreated)="onBlueprintAdded($event)"
32. ></app-cockpit>
34. <hr>
35. <div class="row">
36. <div class="col-md-12">
37. <button class="btn btn-primary" (click)="onChangeFirst()">Change First Element</button>
38. <button class="btn btn-danger" (click)="onDestroyFirst()">Destroy first Component</button>
39. <app-server-element
40. \*ngFor="let serverElement of serverElements"
41. [srvElement]="serverElement"
42. [name]="serverElement.name"
43. >
44. <p #contentParagraph>
45. <strong \*ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>
46. <em \*ngIf="serverElement.type === 'blueprint'" style="color: blue">{{ serverElement.content }}</em>
47. </p>
48. </app-server-element>
49. </div>
50. </div>
51. </div> -->
52. <!-- Unit 7 - Directives Deep Dive; application 4 -->
53. <div class="container">
54. <div class="row">
55. <div class="col-xs-12">
56. <button
57. class = "btn btn-primary"
58. (click) = "onlyOdd = !onlyOdd"
59. >Only show odd numbers</button>
60. <br><br>
62. <ul class="list-group">
63. <div \*ngIf="onlyOdd">
64. <li class="list-group-item"
65. [ngClass]="{odd: odd % 2 !== 0}"
66. [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
67. \*ngFor="let odd of oddNumbers"
68. >{{ odd }}</li>
69. </div>
70. <!-- <div \*ngIf="!onlyOdd">
71. <li class="list-group-item"
72. [ngClass]="{odd: even % 2 !== 0}"
73. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
74. \*ngFor="let even of evenNumbers"
75. >{{ even }}</li>
76. </div>
77. <ng-template [ngIf]="!onlyOdd">
78. <div>
79. <li class="list-group-item"
80. [ngClass]="{odd: even % 2 !== 0}"
81. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
82. \*ngFor="let even of evenNumbers"
83. >{{ even }}</li>
84. </div>
85. </ng-template> -->
86. <div \*appUnless="onlyOdd">
87. <li class="list-group-item"
88. [ngClass]="{odd: even % 2 !== 0}"
89. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
90. \*ngFor="let even of evenNumbers"
91. >{{ even }}</li>
92. </div>
93. </ul>
94. <p appBasicHighlight>Style me with basic directive!</p>
95. <!-- appBetterHighlight -->
96. <p appBetterHighlight [defaultColor]="'yellow'" [highlightColor]="'red'">Style me with a better directive!</p>
97. </div>
98. </div>
99. </div>

5. unless.directive.ts:

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

import { createEmbeddedView } from '@angular/core/src/view/view';

@Directive({

selector: '[appUnless]'

})

export class UnlessDirective {

@Input() set appUnless(condetion:boolean){

if(!condetion){

this.vcRef.createEmbeddedView(this.templateRef);

}else{

this.vcRef.clear();

}

}

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

}

**Section 7: Lecture 93 //Understanding ngSwitch**

1. Before ending this section let’s have a look at another structural directive called ngSwitch.
2. app.component.ts:
3. // import { Component } from '@angular/core';
4. // @Component({
5. // selector: 'app-root',
6. // templateUrl: './app.component.html',
7. // styleUrls: ['./app.component.css']
8. // // styles: [`
9. // // h3{
10. // // color: dodgerblue;
11. // // }
12. // // `]
13. // })
14. // export class AppComponent {
15. // serverElements = [{type: 'server', name: 'Testserver', content: 'Just a test!'}];
16. // name = '';
17. // onServerAdded(serverData: {serverName: string, serverContent: string}){
19. // this.serverElements.push({
20. // type: 'server',
21. // name: serverData.serverName,
22. // content: serverData.serverContent
23. // });
24. // }
26. // onBlueprintAdded(blueprintData: {serverName: string, serverContent: string}){
27. // this.serverElements.push({
28. // type: 'blueprint',
29. // name: blueprintData.serverName,
30. // content: blueprintData.serverContent
31. // });
32. // }
33. // onChangeFirst(){
34. // this.serverElements[0].name = 'Changed!';
35. // }
36. // onDestroyFirst(){
37. // this.serverElements.splice(0,1);
38. // }
39. // }
40. //App Module 7 - Directives Deep Dive
41. import {Component} from '@angular/core'
42. @Component({
43. selector: 'app-root',
44. templateUrl: './app.component.html',
45. styleUrls:['./app.component.css']
46. })
47. export class AppComponent{
48. //numbers = [1,2,3,4,5];
49. oddNumbers = [1,3,5];
50. evenNumbers = [2,4];
51. onlyOdd = false;
52. value = 5;
53. }
54. app.component.html:
55. <!-- Aplication one -->
56. <!--<input type="text" [(ngModel)]="name">
57. <p>{{ name }}</p>-->
58. <!--<div class="container">
59. <div class="row">
60. <div class="col-xs-12">
61. <h3>I am in the AppComponent!!!</h3>-->
63. <!-- <div app-servers></div>-->
64. <!--<app-header></app-header>-->
65. <!--<div class="app-servers"></div>-->
66. <!-- </div>
67. </div>
68. </div>-->
69. <!-- Application 2 -->
70. <!--<app-header></app-header>
71. <div class="container">
72. <div class="row">
73. <div class="col-md-12">
74. <app-recipes></app-recipes>
75. <app-shopping-list></app-shopping-list>
76. </div>
77. </div>
78. </div>-->
79. <!-- Application 3 -->
80. <!-- <div class="container">
81. <app-cockpit
82. (serverCreated)="onServerAdded($event)"
83. (bpCreated)="onBlueprintAdded($event)"
84. ></app-cockpit>
86. <hr>
87. <div class="row">
88. <div class="col-md-12">
89. <button class="btn btn-primary" (click)="onChangeFirst()">Change First Element</button>
90. <button class="btn btn-danger" (click)="onDestroyFirst()">Destroy first Component</button>
91. <app-server-element
92. \*ngFor="let serverElement of serverElements"
93. [srvElement]="serverElement"
94. [name]="serverElement.name"
95. >
96. <p #contentParagraph>
97. <strong \*ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>
98. <em \*ngIf="serverElement.type === 'blueprint'" style="color: blue">{{ serverElement.content }}</em>
99. </p>
100. </app-server-element>
101. </div>
102. </div>
103. </div> -->
104. <!-- Unit 7 - Directives Deep Dive; application 4 -->
105. <div class="container">
106. <div class="row">
107. <div class="col-xs-12">
108. <button
109. class = "btn btn-primary"
110. (click) = "onlyOdd = !onlyOdd"
111. >Only show odd numbers</button>
112. <br><br>
114. <ul class="list-group">
115. <div \*ngIf="onlyOdd">
116. <li class="list-group-item"
117. [ngClass]="{odd: odd % 2 !== 0}"
118. [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
119. \*ngFor="let odd of oddNumbers"
120. >{{ odd }}</li>
121. </div>
122. <!-- <div \*ngIf="!onlyOdd">
123. <li class="list-group-item"
124. [ngClass]="{odd: even % 2 !== 0}"
125. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
126. \*ngFor="let even of evenNumbers"
127. >{{ even }}</li>
128. </div>
129. <ng-template [ngIf]="!onlyOdd">
130. <div>
131. <li class="list-group-item"
132. [ngClass]="{odd: even % 2 !== 0}"
133. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
134. \*ngFor="let even of evenNumbers"
135. >{{ even }}</li>
136. </div>
137. </ng-template> -->
138. <div \*appUnless="onlyOdd">
139. <li class="list-group-item"
140. [ngClass]="{odd: even % 2 !== 0}"
141. [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
142. \*ngFor="let even of evenNumbers"
143. >{{ even }}</li>
144. </div>
145. </ul>
146. <p appBasicHighlight>Style me with basic directive!</p>
147. <!-- appBetterHighlight -->
148. <p appBetterHighlight [defaultColor]="'yellow'" [highlightColor]="'red'">Style me with a better directive!</p>
149. <div [ngSwitch]="value">
150. <p \*ngSwitchCase="5">value is 5</p>
151. <p \*ngSwitchCase="10">value is 10</p>
152. <p \*ngSwitchCase="100">value is 100</p>
153. <p \*ngSwitchDefault>value is Default</p>
154. </div>
155. </div>
156. </div>
157. </div>