

Table of Contents

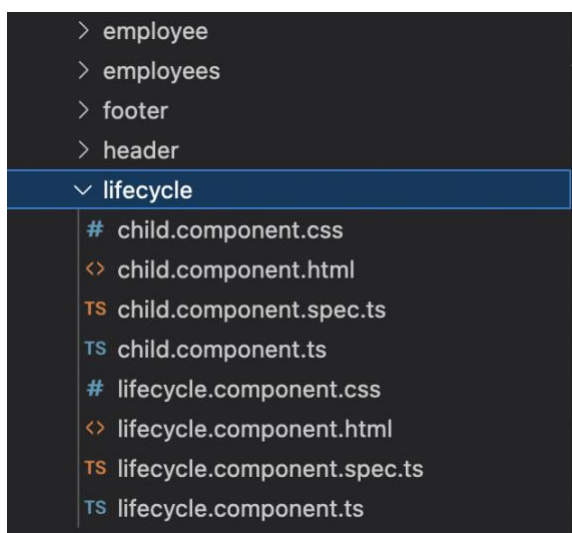
STEP 8: Lifecycle.....	2
STEP 9: Directives.....	5

STEP 8: Lifecycle

1. Add Lifecycle component by executing the below command from the angulardemo folder:
`ng g c lifecycle`
2. Add child component by executing the below command from within the **lifecycle** folder as follows:
`ng g c child --flat`

```
(base) Manishs-MacBook-Pro:angulardemo Shalini$ ng g c lifecycle
Node.js version v17.5.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production.
For more information, please see https://nodejs.org/en/about/releases/.
CREATE src/app/lifecycle/lifecycle.component.css (0 bytes)
CREATE src/app/lifecycle/lifecycle.component.html (24 bytes)
CREATE src/app/lifecycle/lifecycle.component.spec.ts (620 bytes)
CREATE src/app/lifecycle/lifecycle.component.ts (287 bytes)
UPDATE src/app/app.module.ts (894 bytes)
You have new mail in /var/mail/Shalini
(base) Manishs-MacBook-Pro:angulardemo Shalini$ cd src/app/lifecycle/
(base) Manishs-MacBook-Pro:lifecycle Shalini$ ng g c child --flat
Node.js version v17.5.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production.
For more information, please see https://nodejs.org/en/about/releases/.
CREATE src/app/lifecycle/child.component.css (0 bytes)
CREATE src/app/lifecycle/child.component.html (20 bytes)
CREATE src/app/lifecycle/child.component.spec.ts (592 bytes)
CREATE src/app/lifecycle/child.component.ts (271 bytes)
UPDATE src/app/app.module.ts (976 bytes)
```

3. Folder looks as follows:



```
> employee
> employees
> footer
> header
▼ lifecycle
  # child.component.css
  <> child.component.html
  TS child.component.spec.ts
  TS child.component.ts
  # lifecycle.component.css
  <> lifecycle.component.html
  TS lifecycle.component.spec.ts
  TS lifecycle.component.ts
```

4. Comment out the previous tags from app.component.html file and add `<app-lifecycle>` tag

5. Add the following in the lifecycle.component.ts

```
pcountry:string ='usa';
emp = {"name":"Shalini"}
constructor() {
  console.log(`parent constructor`);
}

ngOnInit(): void {
  console.log(`parent ng oninit `);
}
```

6. Add following in lifecycle.component.html

```
<div >
  <h1>parent component! {{pcountry}}</h1>
  <select [(ngModel)]="pcountry" class="form-select mb-5">
    <option value="india">India</option>
    <option value="usa">USA</option>
    <option value="uk">UK</option>
    <option value="ireland">Ireland</option>
  </select>
  <h4>{{emp | json}}</h4>
  <p><input type="text" [(ngModel)]="emp.name" placeholder="Enter
employee name"/></p>
  <p>Show Child : <input type="checkbox" [(ngModel)]="show"/></p>
  <div *ngIf="show">
    <app-child [country]="pcountry" [employee]="emp"></app-child>
  </div></div>
```

7. To pass data from parent to child component use @Input decorator.
Add below lines in child.component.ts file

```
@Input()
country:string ='uk';
@Input()
employee = {"name":""}
```

8. Initialize some dummy data in child.component.ts

```
data:any[] =[
  {"country":"uk","states":["London"]},
  {"country":"india","states":["maharashtra", "UP","MP"]},
  {"country":"ireland","states":["ire1","ire2"]},
  {"country":"usa","states":["Illinois","SFO"]}
]
countrystates:any[]=[]
```

9. Implement the OnInit, OnChanges , DoCheck interface in child.component.ts:
export class ChildComponent implements OnInit, OnChanges , DoCheck,
OnDestroy

10. Override the respective lifecycle methods in child.component.ts

```
ngDoCheck(): void {  
    console.log(`child ng do check ${this.country} : ${this.employee.name}`);  
}  
constructor() {  
    console.log(`child constructor ${this.country}`);  
}  
  
ngOnChanges(changes: SimpleChanges): void {  
    console.log(`child ng on changes ${this.country} : ${this.employee.name}`);  
    this.countrystates = this.data.filter(item=>item.country === this.country)  
  
}  
ngOnInit(): void {  
    console.log(`child ng on init ${this.country}`);  
}  
ngOnDestroy(): void {  
    console.log(`child destroy`);  
}
```

11. Update in child.component.html with below code:

```
<div style="border: 1px solid; padding:20px;">  
<h1>child component!</h1>  
<h3>Country : {{country | uppercase}}</h3>  
<h4 *ngFor="let state of countrystates">{{state.states }}  
</h4>  
<h3>{{employee | json}}</h3>  
</div>
```

12. OnInit and destroy are lifecycle methods that are invoked only once the lifecycle of the component

13. Changing the country in parent will invoke ngOnchanges and ngDoCheck methods

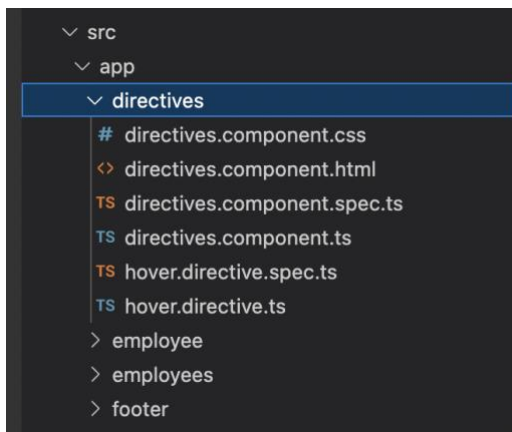
14. Changing the employee name in parent will only invoke ngDoCheck method

STEP 9: Directives

1. Add Directive component by executing the below command from the angulardemo folder:
`ng g c directives`
2. Add a custom directive **from within the directives folder** using the below command:
`ng g d hover`

```
• (base) Manishs-MacBook-Pro:angulardemo Shalini$ ng g c directives
Node.js version v17.5.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production.
For more information, please see https://nodejs.org/en/about/releases/.
CREATE src/app/directives/directives.component.css (0 bytes)
CREATE src/app/directives/directives.component.html (25 bytes)
CREATE src/app/directives/directives.component.spec.ts (627 bytes)
CREATE src/app/directives/directives.component.ts (291 bytes)
UPDATE src/app/app.module.ts (1075 bytes)
You have new mail in /var/mail/Shalini
• (base) Manishs-MacBook-Pro:angulardemo Shalini$ cd src/app/directives/
• (base) Manishs-MacBook-Pro:directives Shalini$ ng g d hover
Node.js version v17.5.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production.
For more information, please see https://nodejs.org/en/about/releases/.
CREATE src/app/directives/hover.directive.spec.ts (220 bytes)
CREATE src/app/directives/hover.directive.ts (139 bytes)
UPDATE src/app/app.module.ts (1158 bytes)
```

3. Folder looks as follows:



4. Comment out the previous tags from app.component.html file and add `<app-directives>` tag
5. To use `ngFor` directive add below code in directives.component.html file:
`<li *ngFor="let pno of [1,2,3,5,7,11,13,17];let i = index">
 {{i}} : {{ pno }} `
6. To use switch case, add below data in directives.component.ts file :
`people: any[] = [
 {
 "name": "Douglas Pace",`

```

"age": 35,
"country": 'MARS'
},
{
"name": "Mcleod Mueller",
"age": 32,
"country": 'USA'
},
{
"name": "Aguirre Ellis",
"age": 34,
"country": 'UK'
},
{
"name": "Cook Tyson",
"age": 32,
"country": 'USA'
}
];

```

7. Display style conditionally as follows:

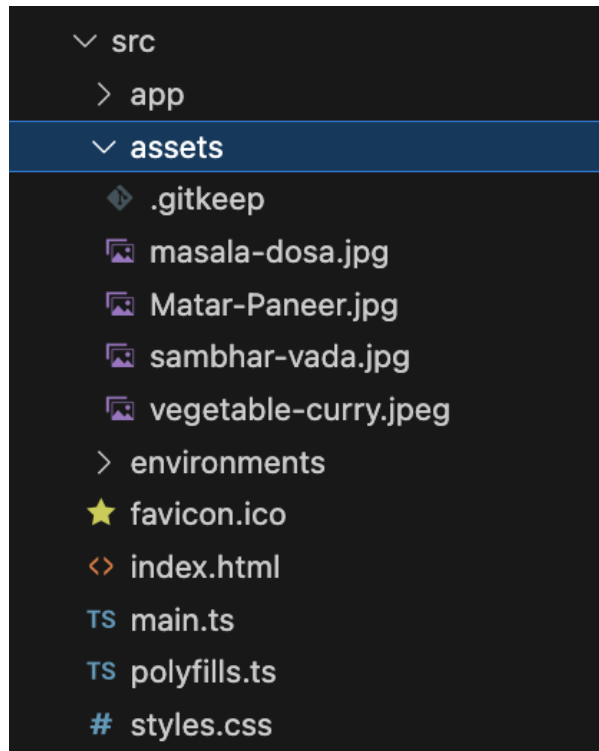
```

<ul *ngFor="let person of people" [ngSwitch]="person.country">
<li *ngSwitchCase="'UK'" class="text-success">
    {{ person.name }} {{ person.country }}
</li>
<li *ngSwitchCase="'USA'" class="text-primary">
    {{ person.name }} {{ person.country }}
</li>
<li *ngSwitchDefault class="text-warning">
    {{ person.name }} {{ person.country }}
</li>
</ul>

```

8. Create custom directives:

- a. This activity needs some dummy images to be added in the assets folder, which is available on the GIT REPO :



- b. Add below property in directives.component.ts file:
itemname:string='masala-dosa.jpg'
- c. Add below code in directives.component.html file:
<h1>Custom Directive</h1>
<select [(ngModel)]="itemname">
 <option value="masala-dosa.jpg">Masala Dosa</option>
 <option value="Matar-Paneer.jpg">Matar Paneer</option>
 <option value="sambhar-vada.jpg">Sambhar Vada</option>
 <option value="vegetable-curry.jpeg">Vegetable Curry</option>
</select>

<p>para </p>
- d. Custom directives are accessed like attributes. To add a border when an image or paragraph is hovered, add below code in constructor of hover.directive.ts. Inject the references of ElementRef and Renderer as follows:
constructor(private el:ElementRef, private render:Renderer2) {
 console.log(el.nativeElement)

```

        render.setStyle(el.nativeElement,'border','2px solid #eee');
        render.setStyle(el.nativeElement,'padding','10px');
    }

```

To use this directive, modify img and p tag of directives.component.html as follows:

```


<p appHover>para </p>

```

- e. To modify the behaviour on mouse over and out, add below listeners in hover.directive.ts file:

```

@HostListener('mouseenter')
mouseenter(){
    this.render.setStyle(this.el.nativeElement,'border','2px solid #56ABEE');
}
@HostListener('mouseleave')
mouseleave(){
    this.render.setStyle(this.el.nativeElement,'border','2px solid #eee');
}

```

- f. To get the dynamic itemname from component add below in hover.directive.ts file:

```

@Input()
itemname:string="";

```

- g. To change the image src attribute dynamically, need to bind this with img tag src attribute. So add below in hover.directive.ts file:

```

@HostBinding("src")
imgSrc:string="";

```

Modify the img tag of directives.component.html as follows:

```



```

- h. To listen for changes add the below lifecycle methods in hover.directive.ts file:

```

// this method listens for only first time and displays the image for the
default itemname
ngOnInit(): void {
    this.imgSrc=`../../assets/${this.itemname}`
}
// this method listens for any subsequent changes and displays the image for
that itemname
ngOnChanges(changes: SimpleChanges): void {
    this.imgSrc=`../../assets/${this.itemname}`
}

```