

## Angular Directives

According to Angular's documentation, a component is really just another form of a directive. There are three types of directives:

### Components

Components are simply directives with their own templates.

### Structural Directives

Structural directives change the DOM layout by adding and removing elements.

### Attribute directives

These directives change the appearance or behavior of a specified element or component.

Since components are technically directives, the "components vs directives" argument gets confusing fast. As best practice, use components when you want to create new DOM elements with their own HTML template. Use attribute directives when you want to change or alter existing DOM elements

### Inbuilt Angular directives

#### Attribute Directives

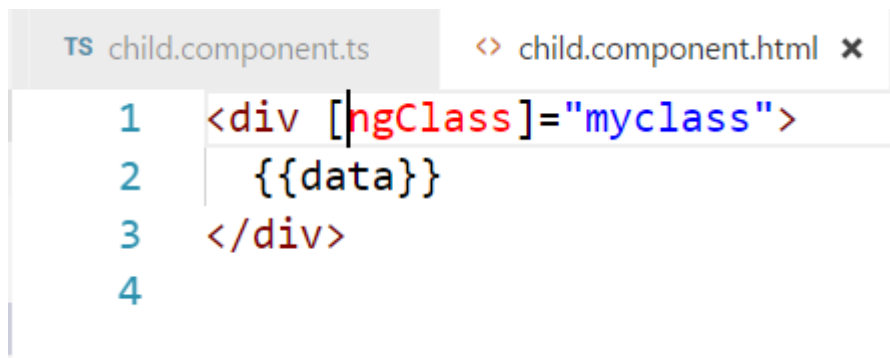
The attribute directive changes the appearance or behavior of a DOM element. These directives look like regular HTML attributes in templates

There are three kinds of angular attribute directives

- Input Directives
- Output Directives
- Input output two way directives

#### Input Directives

All input directives should be enclosed within the symbol []. Some of the examples for input directives are ngStyle and ngClass.



```
TS child.component.ts    <> child.component.html x
1  <div [ngClass]="myclass">
2    {{data}}
3  </div>
4
```

Here myclass is a variable in the class as follows, so when the variable is evaluated appropriate css class is applied to it

```
TS child.component.ts x  <> child.component.html  # child.component.css  <> app
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-child',
5    templateUrl: './child.component.html',
6    styleUrls: ['./child.component.css']
7  })
8  export class ChildComponent implements OnInit {
9
10     data:string="Child Component ....";
11     myclass:string="unit1";
12     constructor() { }
13
14     ngOnInit() {
15     }
16
17 }
18
```

```
TS child.component.ts    <> child.component.html    # child.component.css x    <>
1  .unit1{
2      background-color: ■ rgb(75, 110, 108);
3      color: □ white;
4      padding: 10px;
5      margin: 10px;
6  }
7
8  .unit2{
9      background-color: ■ lightgreen;
10     color: colorblue;
11     padding: 10px;
12     margin: 10px;
13 }
14
```

If we run the component, the style class unit 1 is applied to it.

### Output directive:

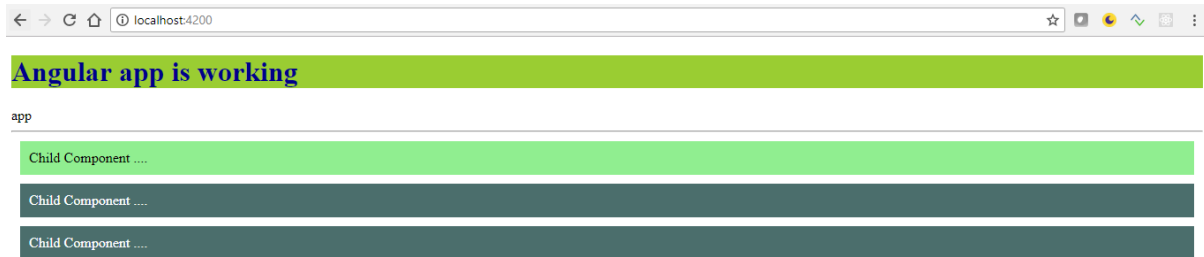
Output directives are generally used for event listening. They are enclosed with in parenthesis ().

```
TS child.component.ts    <> child.component.html x    # child.component.css    <> app.component
1  <div [ngClass]="myclass" (click)="changeBackground()">
2      {{data}}
3  </div>
4
```

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-child',
5    templateUrl: './child.component.html',
6    styleUrls: ['./child.component.css']
7  })
8  export class ChildComponent implements OnInit {
9
10     data:string="Child Component ....";
11     myclass:string="unit1";
12     constructor() { }
13
14     ngOnInit() {
15     }
16     changeBackground(){
17         if(this.myclass=="unit1")
18             this.myclass="unit2"
19         else
20             this.myclass="unit1";
21     }
22
23 }
24
```

So according to code, if we click the class will change and the background color will change appropriately

If we click the first child, we can see the background changed



## Two way bound Directive:

In angular ngModel is two way bound variable where it can be used to bind a variable with a form element . ngModel will work if and only if we include formsModule in the app.module.ts file

```
some # app.component.css TS child.component.ts <> child.component.html TS app.module.ts # child.com
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import {FormsModule} from '@angular/forms';
4
5
6 import { AppComponent } from './app.component';
7 import { ChildComponent } from './child/child.component';
8
9
10 @NgModule({
11   declarations: [
12     AppComponent,
13     ChildComponent
14   ],
15   imports: [
16     BrowserModule,FormsModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
22
```

NgModule(obj?: NgModule): TypeDecorator

Defines an NgModule.

In the app root component, let us have input type as follows and bind it ngModel and lets interpolate the variable in the bottom, As I type in the form the changed values will also gets reflected in the interpolation.

```
1 <h1> Angular app is working </h1>
2
3 {{title}}<br>
4
5 <input type="text" [(ngModel)]="temp">
6 {{temp}}
7
8 <hr>
9 <app-child></app-child>
10 <app-child></app-child>
11 <app-child></app-child>
```

localhost:4200

Angular app is working

app

Typed value Typed value

Child Component ....

Child Component ....

Child Component ....

## Structural Directive

Structural directives are responsible for HTML layout. They shape or reshape the DOM's *structure*, typically by adding, removing, or manipulating elements.

As with other directives, you apply a structural directive to a *host element*. The directive then does whatever it's supposed to do with that host element and its descendants.

Structural directives are easy to recognize. An asterisk (\*) precedes the directive attribute name as in this example.

```

ponent.css    TS child.component.ts    <> child.component.html    TS app.module.ts    # child.component.css    <> app.compone
1    <h1> Angular app is working </h1>
2
3    {{title}}<br>
4
5    <input type="text" [(ngModel)]="temp">
6    {{temp}}
7
8    <h2>Developers</h2>
9    <ul>
10   <li *ngFor="let x of developers">{{x}}</li>
11 </ul>
12
13 <hr>
14 <app-child></app-child>
15 <app-child></app-child>
16 <app-child></app-child>
  
```

In the class, we have a variable call developers which is an array as shown below

```

1    import { Component } from '@angular/core';
2
3    @Component({
4        selector: 'app-root',
5        templateUrl: './app.component.html',
6        styleUrls: ['./app.component.css']
7    })
8    export class AppComponent {
9        title = 'app';
10       developers=["Raghav","Ravi","Ramu"]
11    }
12
  
```

When we render the component it appears like this.





Angular app is working

app

## Developers

- Raghav
- Ravi
- Ramu

|                      |
|----------------------|
| Child Component .... |
| Child Component .... |
| Child Component .... |