

The ngClass Directive

In this lecture, we are going to learn our first directive called **ngClass Directive**.

It's an attribute directive. Therefore, it will change the appearance of an element.

The ngClass Directive allows us to change the classes of an element based on conditions dynamically.

Let's give it a try.

Before we start exploring this directive, we should make some room in our component, it's starting to get cluttered.

Open the app component template file:

At the top of the file, we will add the `<hr>` tag.

Next, open the app stylesheet file:

We will select the `<hr>` tag with some margins to give our template some spacing:

```
# app.css M X
basics > src > app > # app.css > ...
1  input {
2    display: block;
3    margin: 10 0;
4  }
5
6  hr {
7    margin: 20px 0;
8  }
9
```

Let's begin using this directive, for this demonstration, we will create a button for toggling a class.

We should define a class for changing the background and text color of an element. Inside this stylesheet, let's define a class called **blue**:

This class will change the background color to blue and the text color to white.

```
# app.css M X
basics > src > app > # app.css > ...
1  input {
2    display: block;
3    margin: 10 0;
4  }
5
6  hr {
7    margin: 20px 0;
8  }
9
10 .blue {
11   background-color: blue;
12   color: white;
13 }
14
```

Next, open the app component file.

Before we can toggle the class, we should create a property for managing the state of the class. We will create a property called **blueClass**, its initial value will be set to false;

The property we've created will be used as the condition for toggling a class. If the **blueClass** property is set to **false**, the **blueClass** should **not be applied to an element**, vice versa, if the property is set to **true**, we should **load the blue class on the element**.

```

TS app.ts 1, M X
basics > src > app > TS app.ts > ...
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { RouterOutlet } from '@angular/router';
4  import { Post } from '../post/post';
5
6  @Component({
7    selector: 'app-root',
8    imports: [RouterOutlet, Post, CommonModule],
9    templateUrl: './app.html',
10   styleUrls: ['./app.css'],
11 })
12 export class App {
13   protected title = 'basics';
14
15   protected name = 'daniel kandalaft';
16   protected imgUrl = 'https://picsum.photos/id/237/500/500';
17   protected currentDate = new Date();
18   protected cost = 2000;
19   protected temperature = 25.3;
20   protected pizza = {
21     toppings: ['pepperoni', 'bacon'],
22     size: 'large',
23   };
24   blueClass = false; ←
25
26   getName() {
27     return this.name;
28   }
29
30   changeImage(e: KeyboardEvent) {
31     this.imgUrl = (e.target as HTMLInputElement).value;
32   }
33
34   logImg(event: string) {
35     console.log(event);
36   }
37 }
38

```

Let's begin dynamically binding this class to an element.

Switch over to the template, below the <hr> tag add a button element.

Before we apply the **directive**, we should toggle the property, the button we're creating should listen for **click events**. **Bind the click event on the button**. Remember to add the parentheses () around the event name. Otherwise, we won't be able to run an expression. After the event is embedded inside this expression, we will set the **blue class to its opposite value**.

In an earlier section, I mentioned how logic should be outsourced to functions. This recommendation still holds true. However, we are not performing complex logic. Clicking the button should toggle the property nothing more. If we have something as basic as toggling a property, It's much quicker and easier to write it in line. You can consider it an exception to the rule.

The last step is to **dynamically** add the **blue class** to the button. The **ngClass Directive** is designed for **this task**. We can add directives by writing them like attributes. The **blue class** should be added to the button, if the property is set to true. At the moment, we can't bind this class dynamically. If we were to pass in the **blue class**, Angular would add the class to the element. It's the same as using the **class attribute**. **Dynamically Binding Classes to an element requires property binding**, surround the directive with square brackets [], the ngClass directory support various formats, the most common type of format is an object. Object syntax allows us to add multiple classes dynamically. The property name will represent the class we'd like to add to the element. The value will be processed as the condition. Therefore, we can add the blue class to this object, its value will be the blue class property:

```
<button (click)="blueClass = !blueClass" [ngClass]="{ blue: blueClass }">Change</button>
```

If this condition evaluates to true, our blue class will be added to the element. If the class already exists, but the condition evaluates to false, Angular will remove the class. You can add as many classes as you'd like.



```

1  <button (click)="blueClass = !blueClass" [ngClass]="{ blue: blueClass }">
2    Change
3  </button>
4  <hr />
5
6  <input (keyup)="changeImage($event)" [value]="imgURL" />
7
8  <app-post [img]="imgURL" (imgSelected)="logImg($event)">
9    <p>Some caption</p>
10 </app-post>
11
12 <p>Hello {{ name | titlecase }}</p>
13 <p>Hello {{ getName() }}</p>
14 <p>{{ 15 + 13 }}</p>
15 <p>{{ currentDate | date : "MMMM d" }}</p>
16 <p>{{ cost | currency : "JPY" }}</p>
17 <p>{{ temperature | number : "1.0-0" }}</p>
18 <pre>{{ pizza | json }}</pre>
19

```

Let's refresh the page in the browser:

←

→


↻

localhost:4200

Change

← Click

https://picsum.photos/id/237/



Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

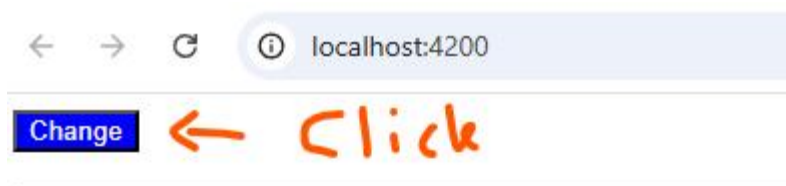
28

July 4

¥2,000

25

```
{  
  "toppings": [  
    "pepperoni",  
    "bacon"  
  ],  
  "size": "large"  
}
```



<https://picsum.photos/id/237/>



Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

28

July 4

¥2,000

25

```
{  
  "toppings": [  
    "pepperoni",  
    "bacon"  
  ],  
  "size": "large"  
}
```



If we click on the button, the blue class gets toggled on the element, it's working perfectly.

The Angular team has introduced the **ngClass Directive** for easily toggling classes on an element.

In the next lecture, we will look at an alternative solution to the **ngClass Directive**.