

The ngFor Directive

In this lecture, we are going to explore one more directive called **ngFor directive**.

It's a directive for **looping an element with an array**. It's common to perform this type of action. If you think about it, a majority of apps made up of lists, a list of users, posts, products, menu items, etc.

Everything we do revolves around lists. As it stands, if we want to output multiple posts, we need to copy and paste the post component. The problem compounds if we need to make adjustments to a specific post. It would be nice if the template could output the current number of posts without updating it as often.

We can cleanly loop through an array by using the **ngFor directive**. This directive will loop through the array in the templates. The element it is applying to, will get output ID on each iteration.

Before we get into this directive, what type of directive do you think the **ngFor directive** is?

Is it an **attribute or structural directive**?

If your guess was a **structural directive**, you'd be correct.

We're modifying the documents elements, not its attributes. **Directives adding or removing elements are considered structural directives.**

For this demonstration, let's try looping through an array of images. We're going to reuse our post component.

First, let's create an array.

Open the app component class file:

Below the **imgURL** property, we will create an array called **images**. To keep it simple, the array will store the same set of images. We will reuse the image we're storing in the **imgURL property**. About three copies should do it.

TS app.ts 1, M X

basics > src > app > TS app.ts > App > getName

```

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   images = [
18     'https://picsum.photos/id/237/500/500',
19     'https://picsum.photos/id/237/500/500',
20     'https://picsum.photos/id/237/500/500',
21   ];
22   protected currentDate = new Date();
23   protected cost = 2000;
24   protected temperature = 25.3;
25   protected pizza = {
26     toppings: ['pepperoni', 'bacon'],
27     size: 'large',
28   };
29   blueClass = false;
30   fontSize = 16;
31
32   getName() {
33     return this.name;
34   }
35
36   changeImage(e: KeyboardEvent) {
37     this.imgURL = (e.target as HTMLInputElement).value;
38   }
39
40   logImg(event: string) {
41     console.log(event);
42   }
43 }
44

```



Next, let's open the app template file.

On the post component, we're going to add the **ngFor directive with the asterisk * character**. As a reminder, we should always add the asterisk * character before the name of the directive. It's a shortcut for wrapping the element with **ng-template element**. It's much cleaner.

Moving on, the value of this directive is vastly different from other directives throughout this section. We've been **binding directives to properties or objects**. Unlike other directives, the **ngFor directive has a special set of syntax rules**. Let's write it first, then we will review the syntax:

```
<app-post *ngFor="let image of images" [img]="image" (imgSelected)="logImg($event)"><p>Some  
caption</p></app-post>
```

This syntax can be broken into two pieces:

- In the very beginning, we've written **let image**. The **let keyword** is similar to the **let keyword** in JavaScript. It allows us to create a variable. In this example, we are creating a variable called image.
- Next, we need to provide an array, we can specify the array by adding the **of keyword**. Afterwards, **we can reference the array on each iteration**. Angular will assign the current element in the iteration to the image variable. In this example, the image variable will hold the current post in the iteration.
- We should update our component to accept the new data. Currently, the component will throw an error. Instead of passing in the **imgURL property**, we can pass in the **image variable**. The **image variable** is accessible in our loop.

This includes the component itself.

```

<> app.html M X
basics > src > app > <> app.html > ...
  Go to component
1  <!-- <ng-template [ngIf]="blueClass">
2    <p>Button is blue.</p>
3  </ng-template> -->
4
5  <p *ngIf="blueClass">Button is blue.</p>
6
7  <button
8    (click)="blueClass = !blueClass"
9    [ngClass]="{ blue: blueClass }"
10   [ngStyle]="{ 'font-size.px': fontSize }"
11 >
12   Change
13 </button>
14 <hr />
15
16 <input (keyup)="changeImage($event)" [value]="imgURL" />
17
18 <app-post
19   *ngFor="let image of images"
20   [img]="image"
21   (imgSelected)="logImg($event)"
22 >
23   <p>Some caption</p>
24 </app-post>
25
26 <p>Hello {{ name | titlecase }}</p>
27 <p>Hello {{ getName() }}</p>
28 <p>{{ 15 + 13 }}</p>
29 <p>{{ currentDate | date : "MMMM d" }}</p>
30 <p>{{ cost | currency : "JPY" }}</p>
31 <p>{{ temperature | number : "1.0-0" }}</p>
32 <pre>{{ pizza | json }}</pre>
33

```

Therefore, we can pass on the image variable to the component. It's much more efficient to use the **ngFor directive**. We don't have to copy and paste the directive multiple times. Angular will always output the correct number of posts. Nor more, no less.

Before you verify if our loop is working, we should verify the image is unique. At the moment, we will see the same image and cation. Fortunately, we can access the index of the current loop iteration.

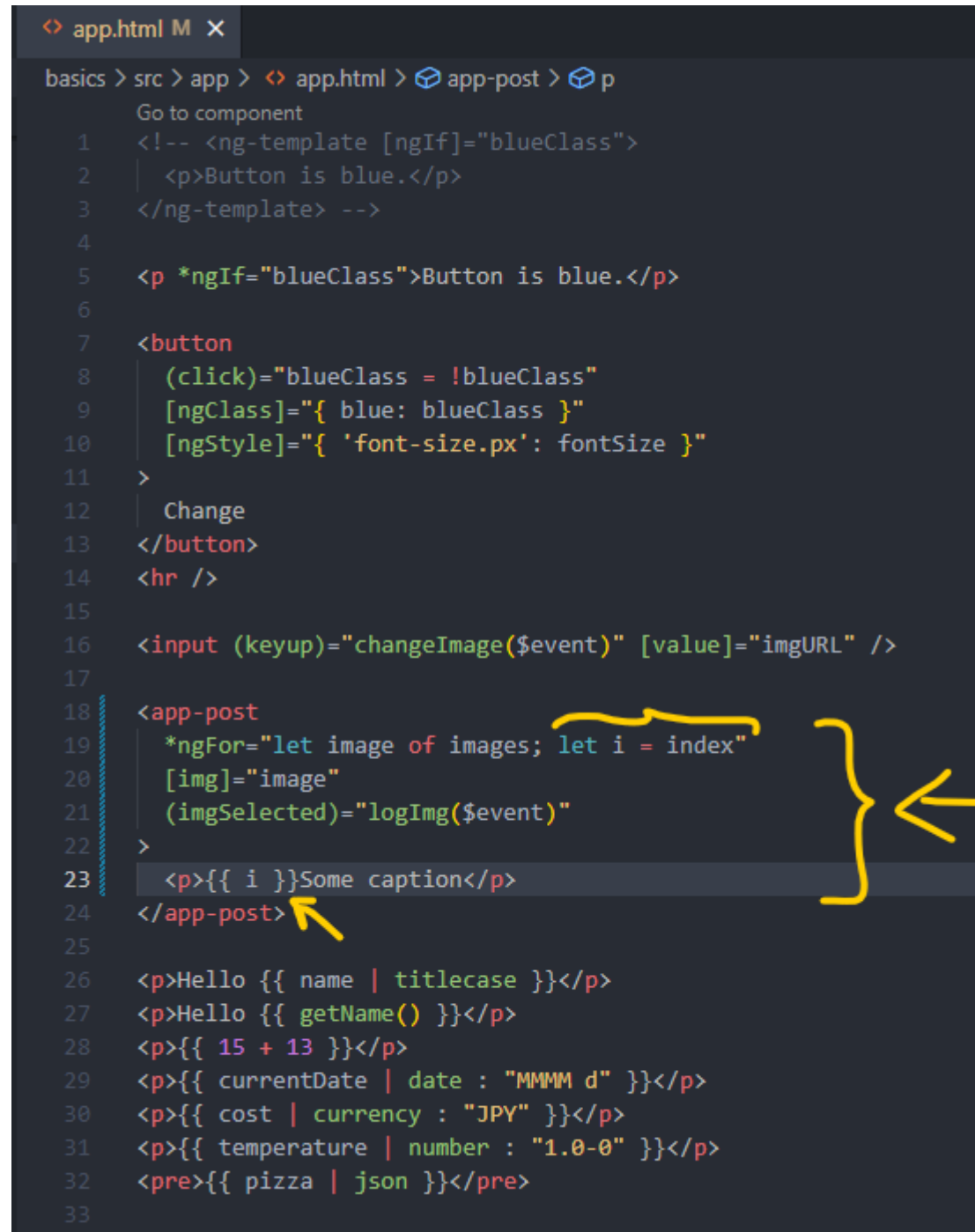
Inside the **ngFor directive**, we will add the following: **let i = index**.

We're creating another variable called i.

It's being assigned to the **index property from the ngFor directive by storing the index in a variable**. We will be able to reference it inside our template.

Let's try outputting the index.

Inside the caption, we will add an expression for the i variable:



```

<> app.html M X
basics > src > app > <> app.html > app-post > p
  Go to component
  1 <!-- <ng-template [ngIf]="blueClass">
  2 | <p>Button is blue.</p>
  3 </ng-template> -->
  4
  5 <p *ngIf="blueClass">Button is blue.</p>
  6
  7 <button
  8 |   (click)="blueClass = !blueClass"
  9 |   [ngClass]="{ blue: blueClass }"
 10 |   [ngStyle]="{ 'font-size.px': fontSize }"
 11 >
 12 |   Change
 13 </button>
 14 <hr />
 15
 16 <input (keyup)="changeImage($event)" [value]="imgURL" />
 17
 18 <app-post
 19 |   *ngFor="let image of images; let i = index"
 20 |   [img]="image"
 21 |   (imgSelected)="logImg($event)"
 22 >
 23 |   <p>{{ i }}Some caption</p>
 24 </app-post>
 25
 26 <p>Hello {{ name | titlecase }}</p>
 27 <p>Hello {{ getName() }}</p>
 28 <p>{{ 15 + 13 }}</p>
 29 <p>{{ currentDate | date : "MMMM d" }}</p>
 30 <p>{{ cost | currency : "JPY" }}</p>
 31 <p>{{ temperature | number : "1.0-0" }}</p>
 32 <pre>{{ pizza | json }}</pre>
 33
  
```

As desired, the image has been rendered three times. On each iteration, the caption will contain the index of the current iteration. Since arrays are zero indexed, the first image will have an index of zero.

0



← → ↻ ⓘ localhost:4200

Change

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



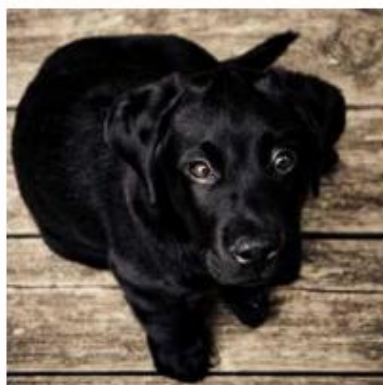
0Some caption



1



1Some caption



2



2Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

28

July 5

¥2,000

25

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

We've successfully looped through the array.

We're finished with learning directives. It's time to move on to the actual project.