The ngIf Directive

In this lecture, we are going to **dynamically render content with the ngIf Directive.**

This directive is considered a **structural directive.**

A structural directive can add or remove elements from the document.

This is completely different from attribute directives.

Attribute directives will modify an attribute appearance through attributes. Whereas structural directives are more powerful, they can manipulate elements by adding or removing them.

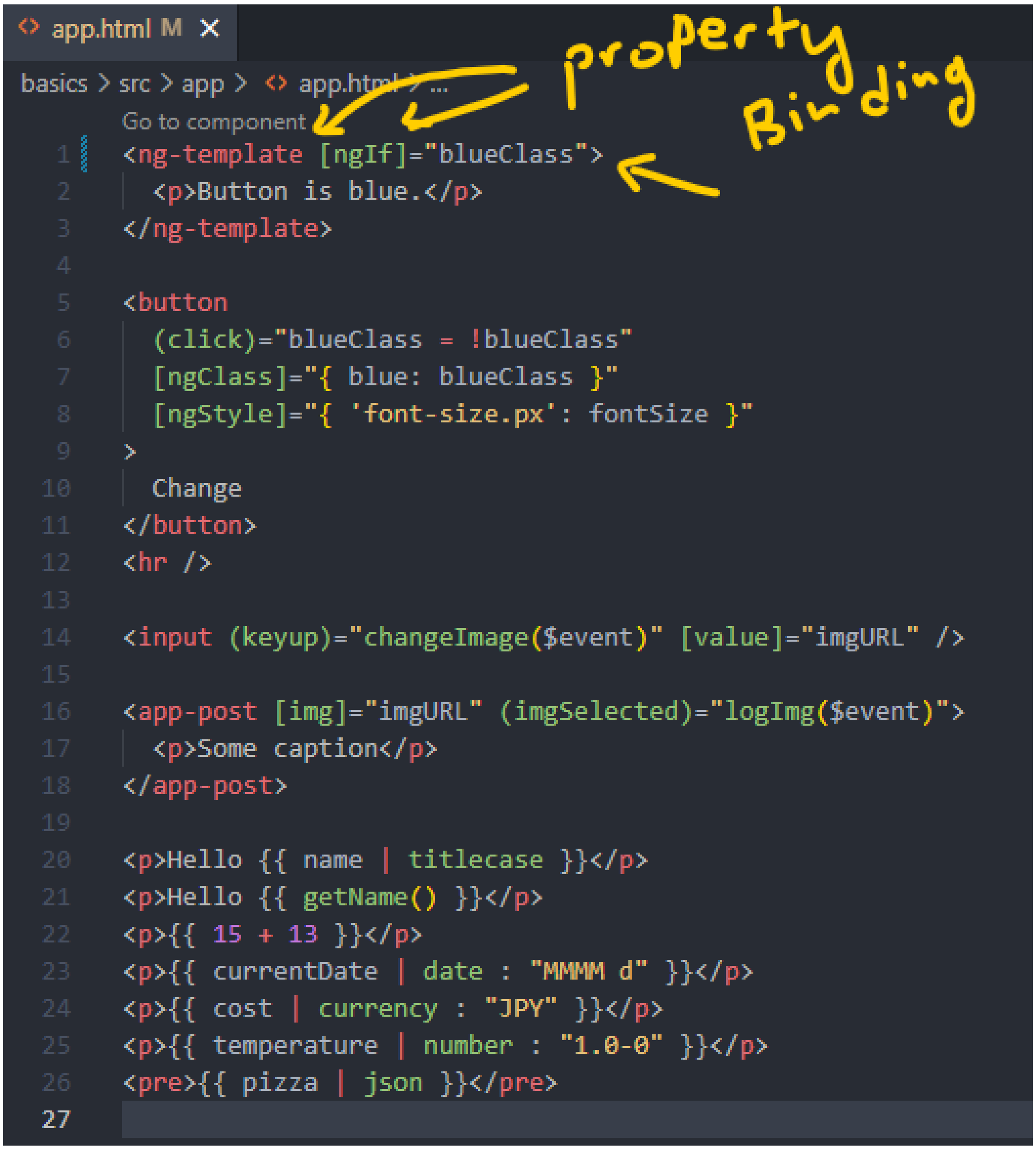
The ngIf directive works similarly to an if statement in JavaScript. It'll execute code if a condition is true.

In this case, we will display a piece of the template based on a condition. Therefore, the **ngIf directive is a structural directive.**

We will continue working on the app template file:

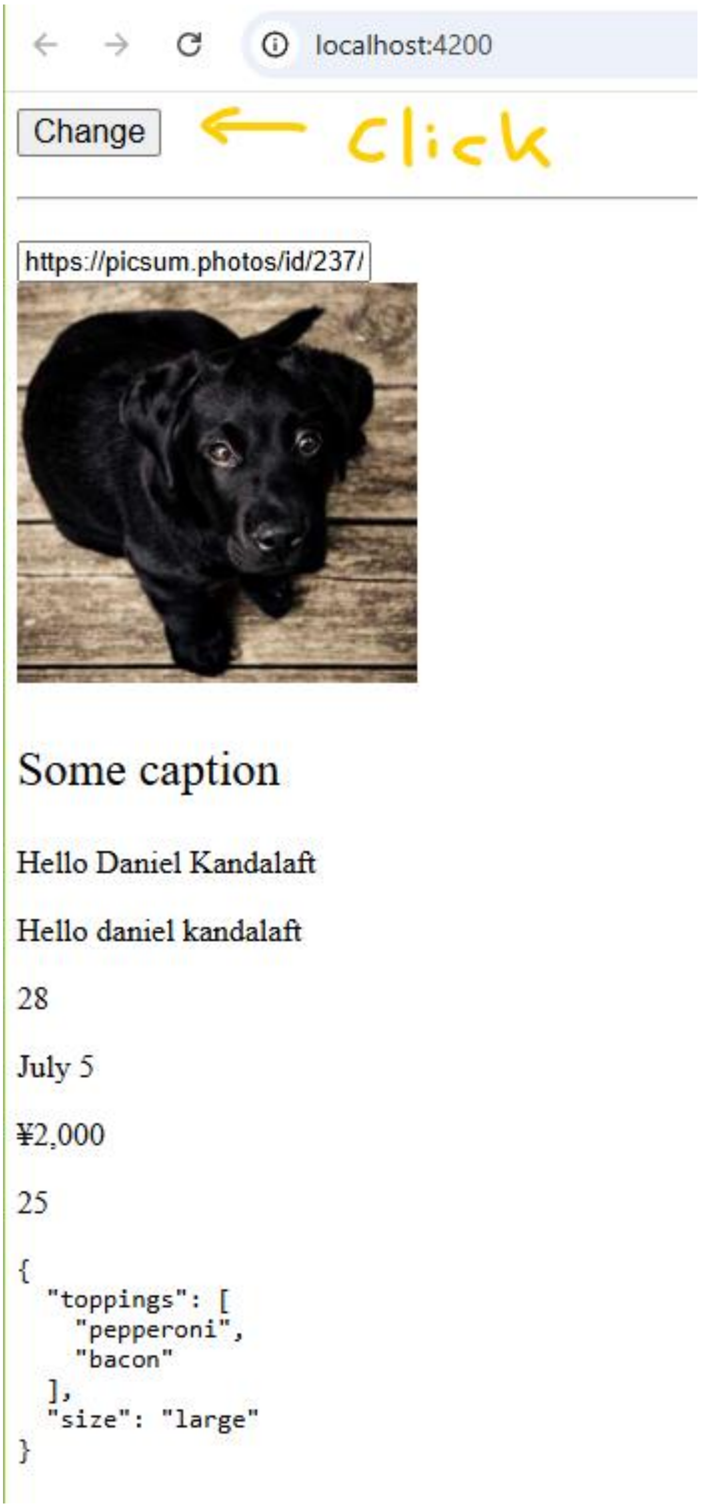
Structural directives must be applied to ng-template elements.

Let's add the **ngIf directive** to the element [**ng-template**]. Just like **attribute directive**, **structural directives** should be wrapped with **square brackets for property binding []. Otherwise, we won't be able to use an expression as a value.** So, let's bind this directive to the **blueClass property**. As a reminder, this property will hold a Boolean value for determining if a class should be added to a button. We are going to be using the same condition. If a button is blue, the message should appear:



```
<ng-template [ngIf]="blueClass">
  <p>Button is blue.</p>
</ng-template>

<button
  (click)="blueClass = !blueClass"
  [ngClass]="{ blue: blueClass }"
  [ngStyle]="{ 'font-size.px': fontSize }"
>
  Change
</button>
<hr />

<input (keyup)="changeImage($event)" [value]="imgURL" />

<app-post [img]="imgURL" (imgSelected)="logImg($event)">
  <p>Some caption</p>
</app-post>

<p>Hello {{ name | titlecase }}</p>
<p>Hello {{ getName() }}</p>
<p>{{ 15 + 13 }}</p>
<p>{{ currentDate | date : "MMMM d" }}</p>
<p>{{ cost | currency : "JPY" }}</p>
<p>{{ temperature | number : "1.0-0" }}</p>
<pre>{{ pizza | json }}</pre>
```

Let’s check out the app in the browser:



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

Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

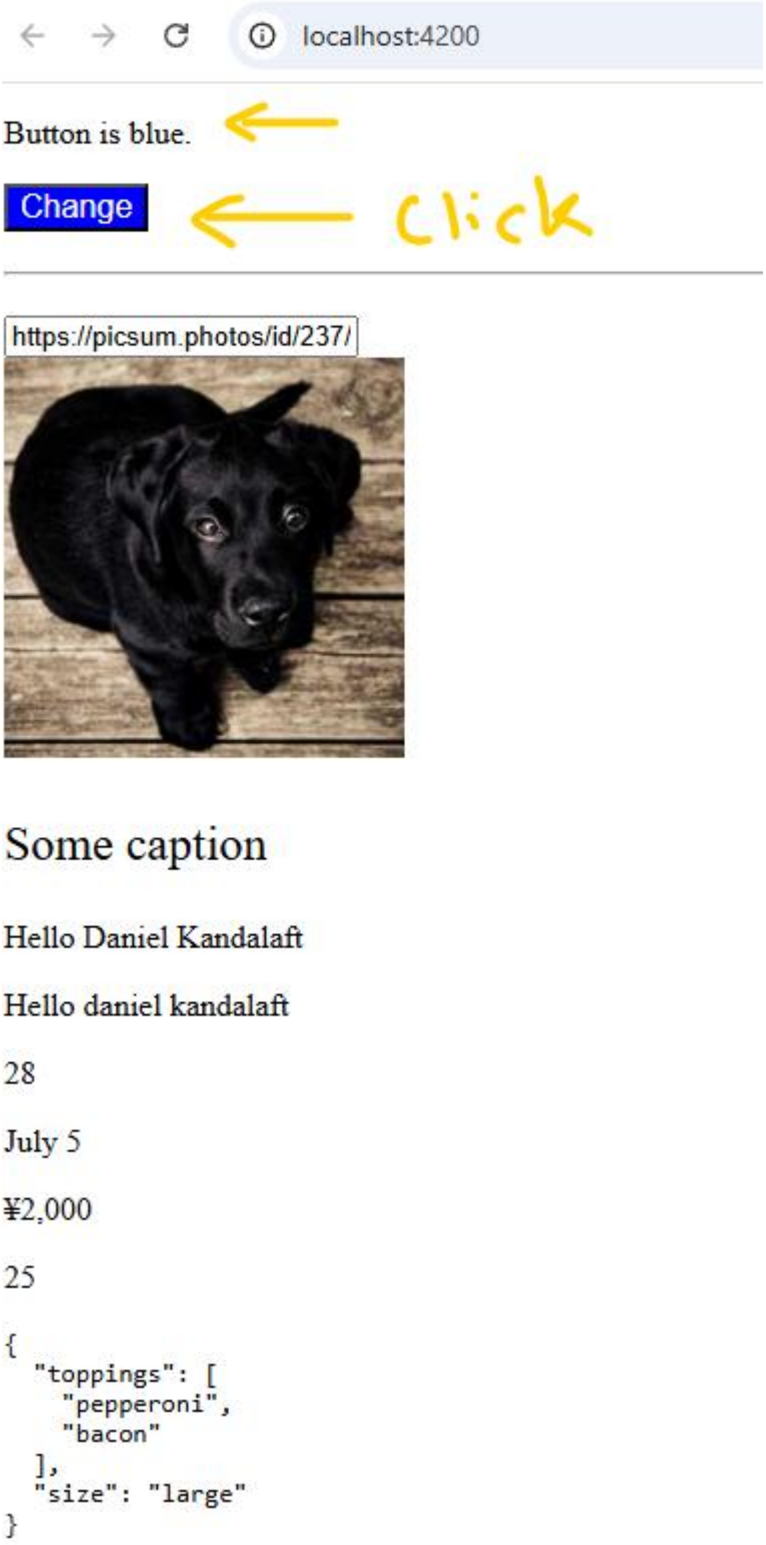
28

July 5

¥2,000

25

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



Button is blue.

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

Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

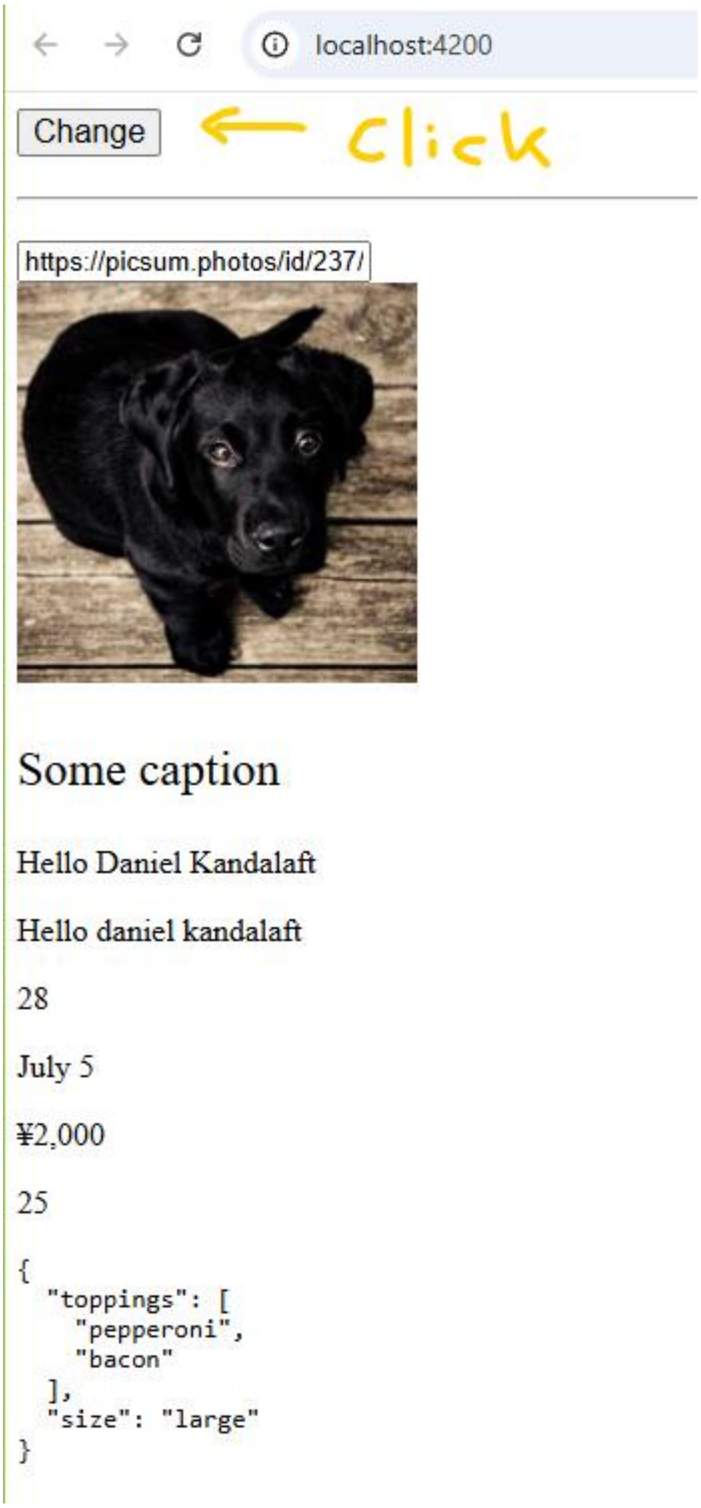
28

July 5

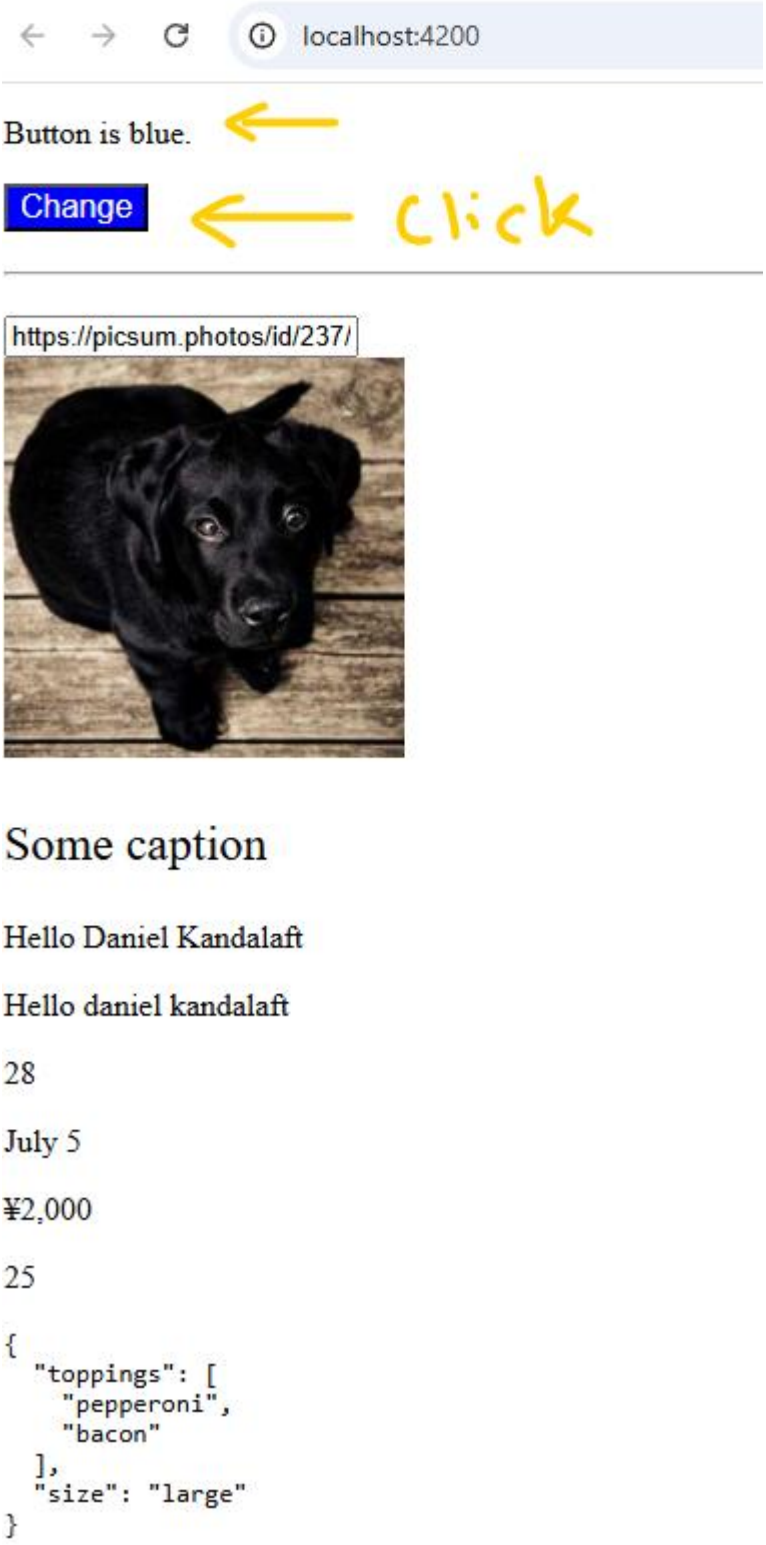
¥2,000

25

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



```
Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

28

July 5

¥2,000

25

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



Button is blue.

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

Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

28

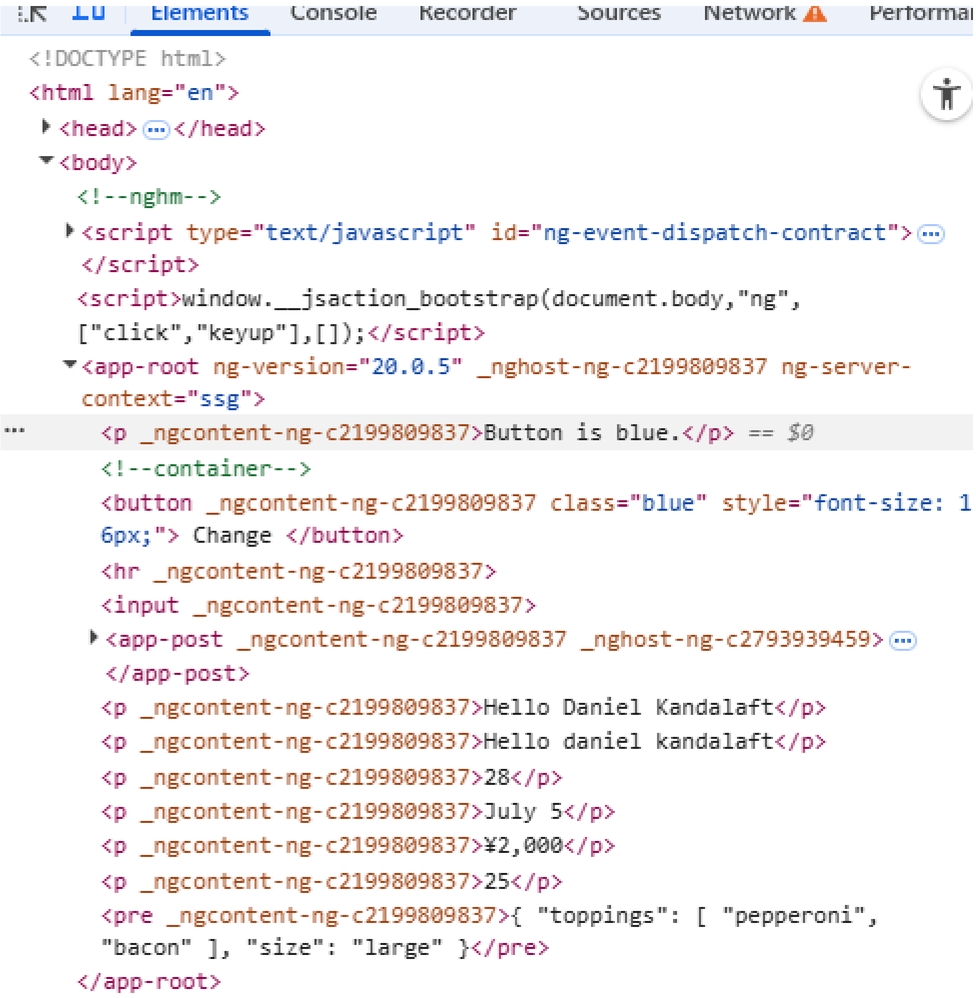
July 5

¥2,000

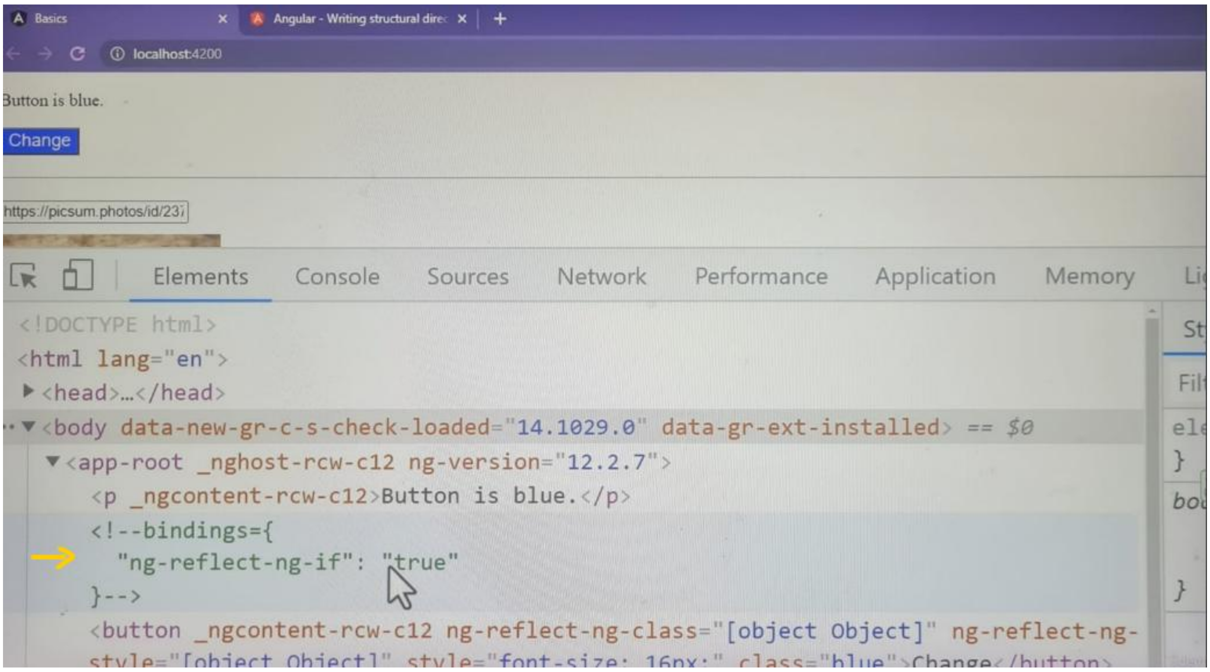
25

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

If we press the button, the message appears. In the developer tools, we can clearly see the elements. The **ng-template element** is not present in the document:



```
<!DOCTYPE html>
<html lang="en">
 <head> … </head>
 <body>
   <!--nghm-->
  <script type="text/javascript" id="ng-event-dispatch-contract"> …
   </script>
   <script>window.__jsaction_bootstrap(document.body,"ng",
   ["click","keyup"],[]);</script>
  <app-root ng-version="20.0.5" _nghost-ng-c2199809837 ng-server-
   context="ssg">
     <p _ngcontent-ng-c2199809837>Button is blue.</p> == $0
     <!--container-->
     <button _ngcontent-ng-c2199809837 class="blue" style="font-size: 1
     6px;"> Change </button>
     <hr _ngcontent-ng-c2199809837>
     <input _ngcontent-ng-c2199809837>
    <app-post _ngcontent-ng-c2199809837 _nghost-ng-c2793939459> …
     </app-post>
     <p _ngcontent-ng-c2199809837>Hello Daniel Kandalaft</p>
     <p _ngcontent-ng-c2199809837>Hello daniel kandalaft</p>
     <p _ngcontent-ng-c2199809837>28</p>
     <p _ngcontent-ng-c2199809837>July 5</p>
     <p _ngcontent-ng-c2199809837>¥2,000</p>
     <p _ngcontent-ng-c2199809837>25</p>
     <pre _ngcontent-ng-c2199809837>{ "toppings": [ "pepperoni",
     "bacon" ], "size": "large" }</pre>
   </app-root>
```



```
<!DOCTYPE html>
<html lang="en">
 <head>…</head>
 <body data-new-gr-c-s-check-loaded="14.1029.0" data-gr-ext-installed> == $0
   <app-root _nghost-rcw-c12 ng-version="12.2.7">
     <p _ngcontent-rcw-c12>Button is blue.</p>
     <!--bindings={
       "ng-reflect-ng-if": "true"
     }-->
     <button _ngcontent-rcw-c12 ng-reflect-ng-class="[object Object]" ng-reflect-ng-
     style="[object Object]" style="font-size: 16px;" class="blue">Change</button>
```

Angular will take care of removing the **ng-template element** for us. Thus, we don't have to worry about ruining the integrity of our document, chich could lead to CSS errors. We can continue to act as if it was never there. However, it can be distracting to write in our templates. **We have to**

constantly write the ng-template element whenever we want to use a structural directive. Luckily, we can avoid it altogether with **shorthand syntax.**

Switch back to the editor.

We're going to move the **ngIf directive** form the **ng-template element** to the paragraph element.

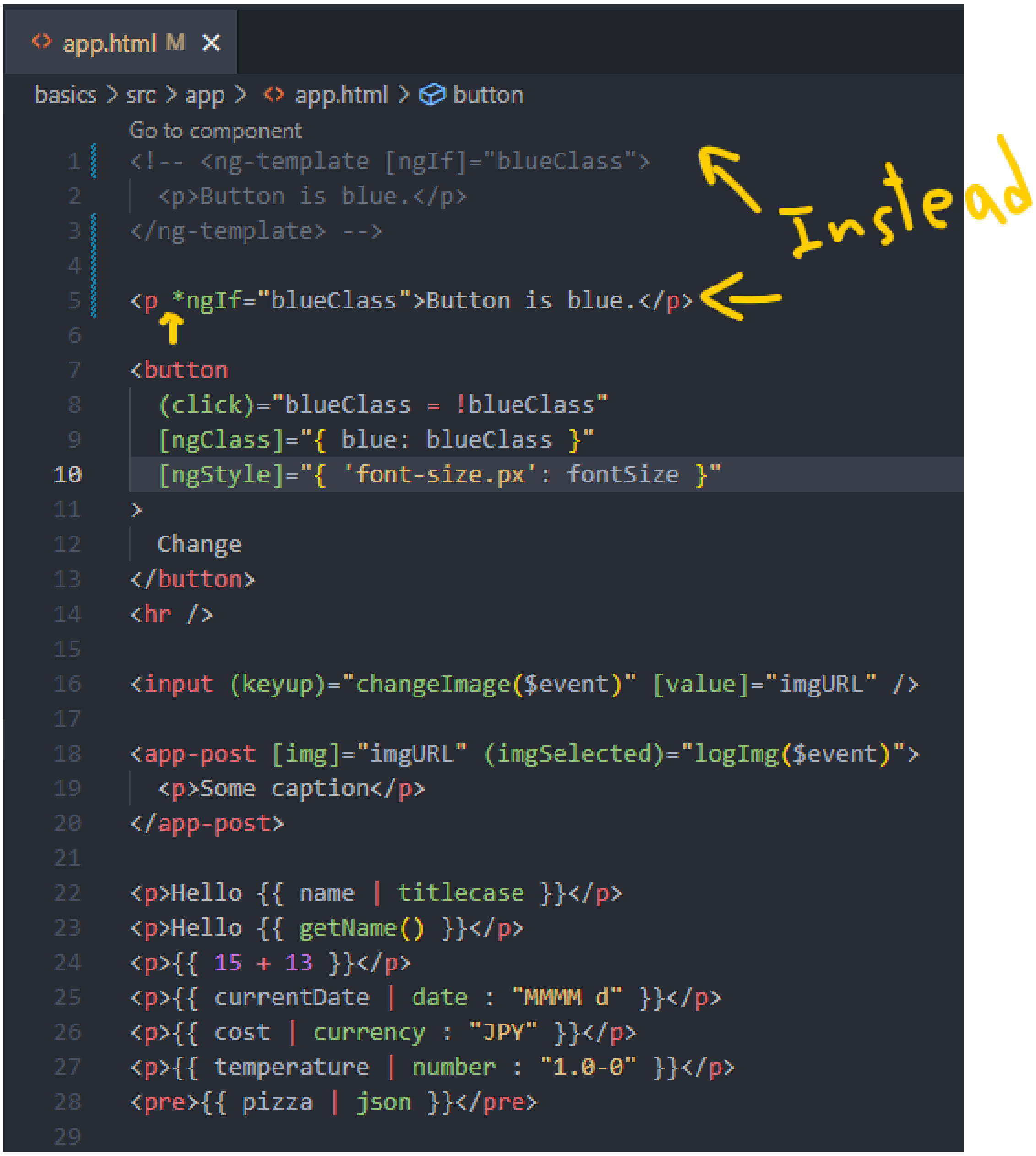
Next, we will remove the **ng-template element**.

Lastly, we will replace these **square brackets []** with an **asterisk character * before the directive name**. This character * can be added to **structural directives**. It doesn't just apply to the **ngIf directive. By adding this character, Angular will perform two actions:**

1. It'll wrap the element with the **ng-template element.**
2. The value will be **interpreted as an expression. We don't have to wrap the directive with square brackets [].**

This syntax is a shorthand way of writing structural directives. It also makes it easy to identify structural directives.

<p *ngIf="blueClass">Button is blue.</p>



```
<!-- <ng-template [ngIf]="blueClass">
  <p>Button is blue.</p>
</ng-template> -->

<p *ngIf="blueClass">Button is blue.</p>

<button
  (click)="blueClass = !blueClass"
  [ngClass]="{ blue: blueClass }"
  [ngStyle]="{ 'font-size.px': fontSize }"
>
  Change
</button>
<hr />

<input (keyup)="changeImage($event)" [value]="imgURL" />

<app-post [img]="imgURL" (imgSelected)="logImg($event)">
  <p>Some caption</p>
</app-post>

<p>Hello {{ name | titlecase }}</p>
<p>Hello {{ getName() }}</p>
<p>{{ 15 + 13 }}</p>
<p>{{ currentDate | date : "MMMM d" }}</p>
<p>{{ cost | currency : "JPY" }}</p>
<p>{{ temperature | number : "1.0-0" }}</p>
<pre>{{ pizza | json }}</pre>
```

For the rest of this course, we will use this shorthand syntax whenever writing structural directives.

Link to a page of shorthand examples: https://v17.angular.io/guide/structural-directives#shorthand-examples

https://angular.dev/guide/directives/structural-directives#shorthand-examples

Shorthand examples

The following table provides shorthand examples:

Shorthand	How Angular interprets the syntax
`*myDir="let item of [1,2,3]"`	`<ng-template myDir let-item [myDirOf]="[1, 2, 3]">`

Shorthand	How Angular interprets the syntax
`*myDir="let item of [1,2,3] as items; trackBy: myTrack; index as i"`	`<ng-template myDir let-item [myDirOf]="[1,2,3]" let-items="myDirOf" [myDirTrackBy]="myTrack" let-i="index">`
`*ngComponentOutlet="componentClass";`	`<ng-template [ngComponentOutlet]="componentClass">`
`*ngComponentOutlet="componentClass; inputs: myInputs";`	`<ng-template [ngComponentOutlet]="componentClass" [ngComponentOutletInputs]="myInputs">`
`*myDir="exp as value"`	`<ng-template [myDir]="exp" let-value="myDir">`

Angular provides examples of how **structural directives** are **transformed** into the **ng-template element** with this **shorthand syntax.**

In the next lecture, we will explore another **structural directive**.