

Understanding ng-template

In this lecture, we are going to talk about the **ng-template element for understanding Structural Directives**.

Under the hood, Angular relies on the **ng-template element for rendering content manipulated by a structural directive. It's a special element not recognized by the browser.**

The purpose of the ng-template element is to store one or more elements in memory.

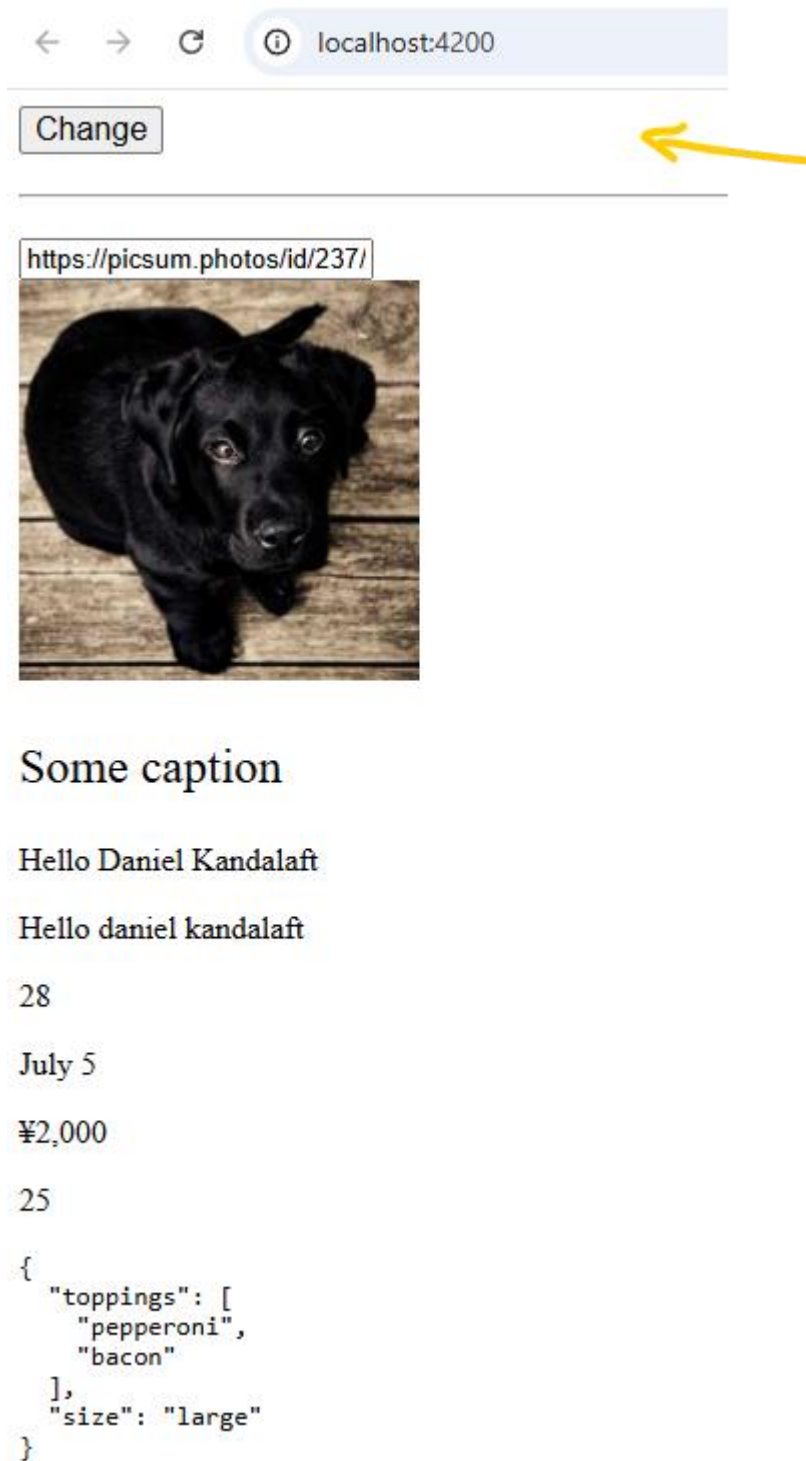
Let's explore what that means.

For this example, we will need to open the app template file.

At the top of the file, we will add the **ng-template element**.

Next, let's insert a paragraph with some text. The text will allow the user to know if the button is blue. There's one thing to understand about this element. **Angular will not display the contents of this element in the browser. It'll detect we have an ng-template element and immediately hide it. The contents of the template will be stored in memory, a.k.a. a variable.** We can verify this behaviour by switching to the browser.

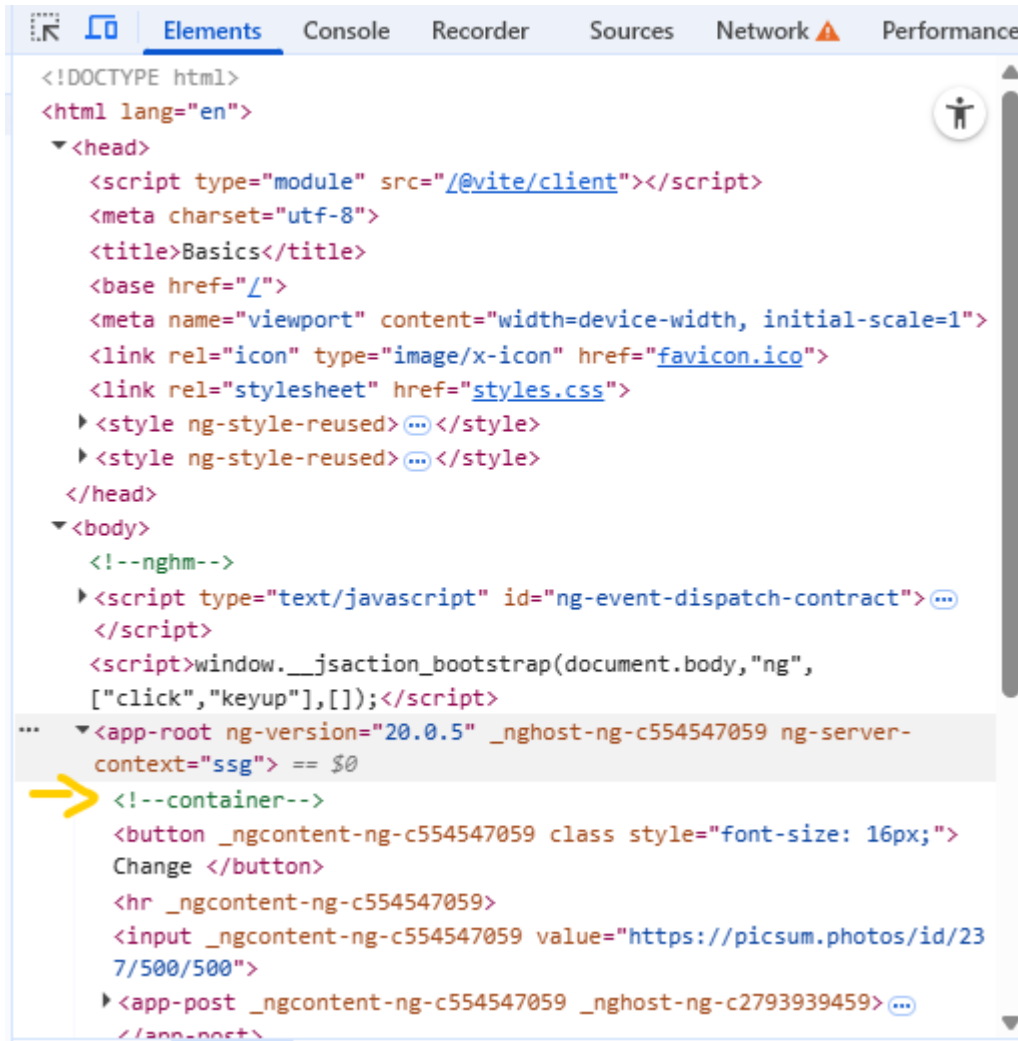
```
app.html M X
basics > src > app > app.html > ng-template > p
Go to component
1 <ng-template>
2   <p>Button is blue.</p>
3 </ng-template>
4
5 <button
6   (click)="blueClass = !blueClass"
7   [ngClass]="{ blue: blueClass }"
8   [ngStyle]="{ 'font-size.px': fontSize }"
9 >
10   Change
11 </button>
12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



As you can see, the message is gone. That leads us to the question, why does Angular hide this element?

In some cases, we may not want to display content immediately. Instead, we may want to render content based on a condition. We need a way to tell Angular which pieces of content should be conditionally rendered. By wrapping the content with the **ng-template element**, **Angular will not render the content, but still be aware of it. In its place, the content will be replaced with an HTML comment.**

Let's open the elements panel in the Developers Tools. Inside the app components, we will find a comment that says: **Container**. Angular has generated this comment. It is a temporary placeholder. If we decide to render the content, Angular will add our content next to this comment.



```

<!DOCTYPE html>
<html lang="en">
  <head>
    <script type="module" src="/@vite/client"></script>
    <meta charset="utf-8">
    <title>Basics</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link rel="stylesheet" href="styles.css">
    <style ng-style-reused>...</style>
    <style ng-style-reused>...</style>
  </head>
  <body>
    <!--nghtm-->
    <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" _ngghost-ng-c554547059 ng-server-
    context="ssg"> == $0
    <!--container-->
    <button _ngcontent-ng-c554547059 class style="font-size: 16px;">
    Change </button>
    <hr _ngcontent-ng-c554547059>
    <input _ngcontent-ng-c554547059 value="https://picsum.photos/id/23
    7/500/500">
    <app-post _ngcontent-ng-c554547059 _ngghost-ng-c2793939459>...
    </app-post>
  </body>
</html>

```

The next step is to render the content based on a condition. This is where **structural directives** come into play.

We will handle rendering our content in the next lecture.