Content Projection

In this lecture, we're going to add **Content Projection** to our **post component**, the post component displays an image.

Captions can accompany images. We should add an option for adding captions to our images. The captions should come from the parent component.

So far, we've learned how to pass down data from the parent component to the child component. We can add an @Input decorator to a property in the post class to allow the parent component to configure it. However, it may not always be the best way to pass down content.

What if we want to pass down HTML content?

Our caption can be as simple as a paragraph tag to a complex structure of elements. We can use inputs to accomplish this task, but I'm not a fan of writing HTML inside my classes.

Instead of using inputs, we can accomplish this task with **Content Projection.** It sounds scary, but I promise it's not.

Content Projection is the process of loading **content inserted into a components tag.**

Let's open the app component template file:

Components are written with the opening and closing tags like other HTML elements, elements defined by the browser can have **content inserted in them**. For example, our paragraph tags have content. If we want, we can add more tags. This will create a **nested structure**. We can do the same thing with our **custom components.**

Components receiving content can load content anywhere in the template by default. **Components will ignore the content inserted in between the tags.** We need to tell Angular where to insert the contents. This process is called **Content Projection**.

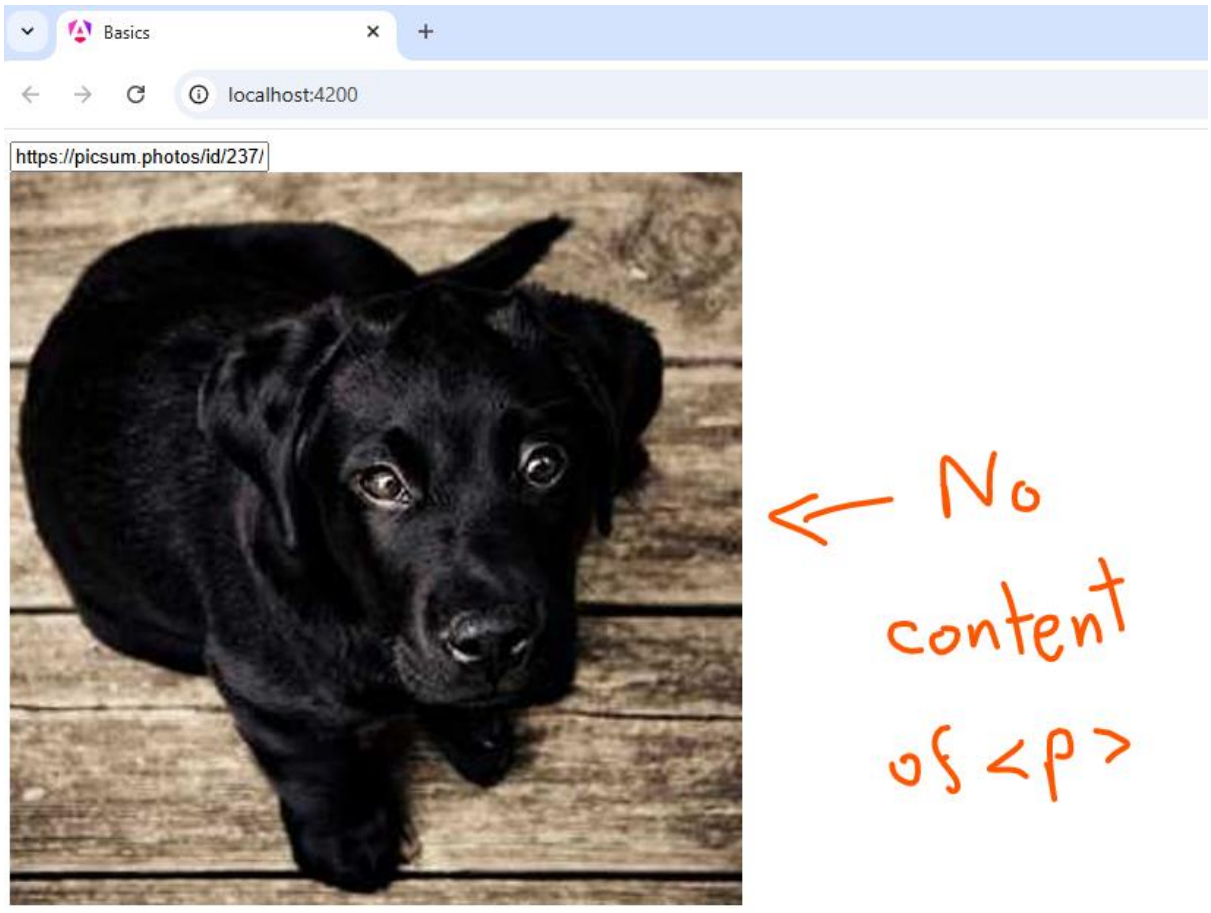
Let's pass in a pair of paragraph tags with some dummy content inside the post component:

app.html M

basics > src > app > app.html > ...

Go to component

```
1  <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3  <app-post [img]="imgURL" (imgSelected)="logImg($event)">
4    <p>Some caption</p>
5  </app-post>
6
7  <p>Hello {{ name.toLocaleUpperCase() }}</p>
8  <p>Hello {{ getName() }}</p>
9  <p>{{ 15 + 13 }}</p>
10
```

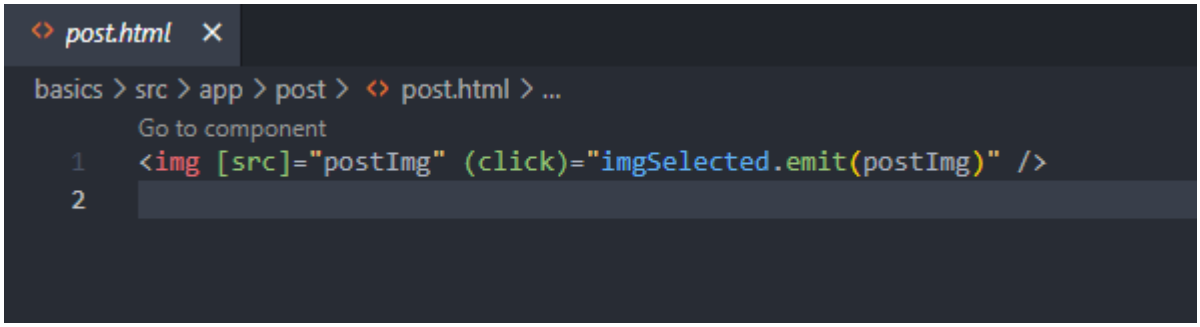


Hello LUIS

Hello Luis

28

Next, open the post template file:

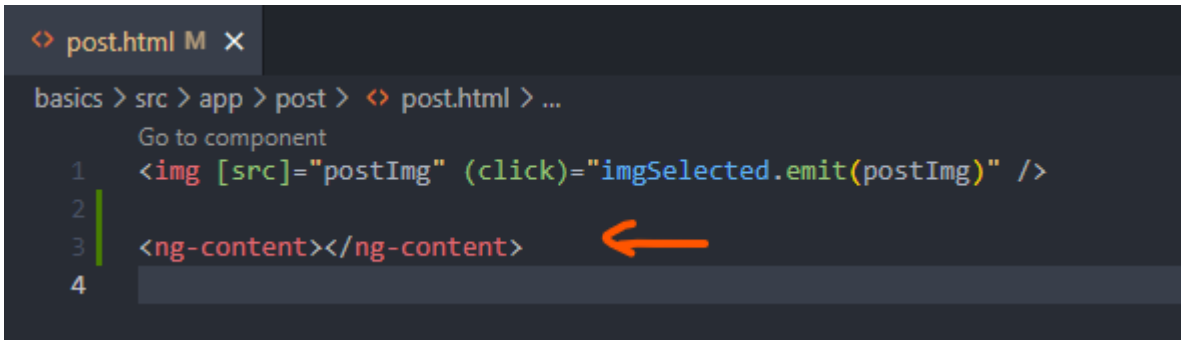


We need to tell Angular where to load the content.

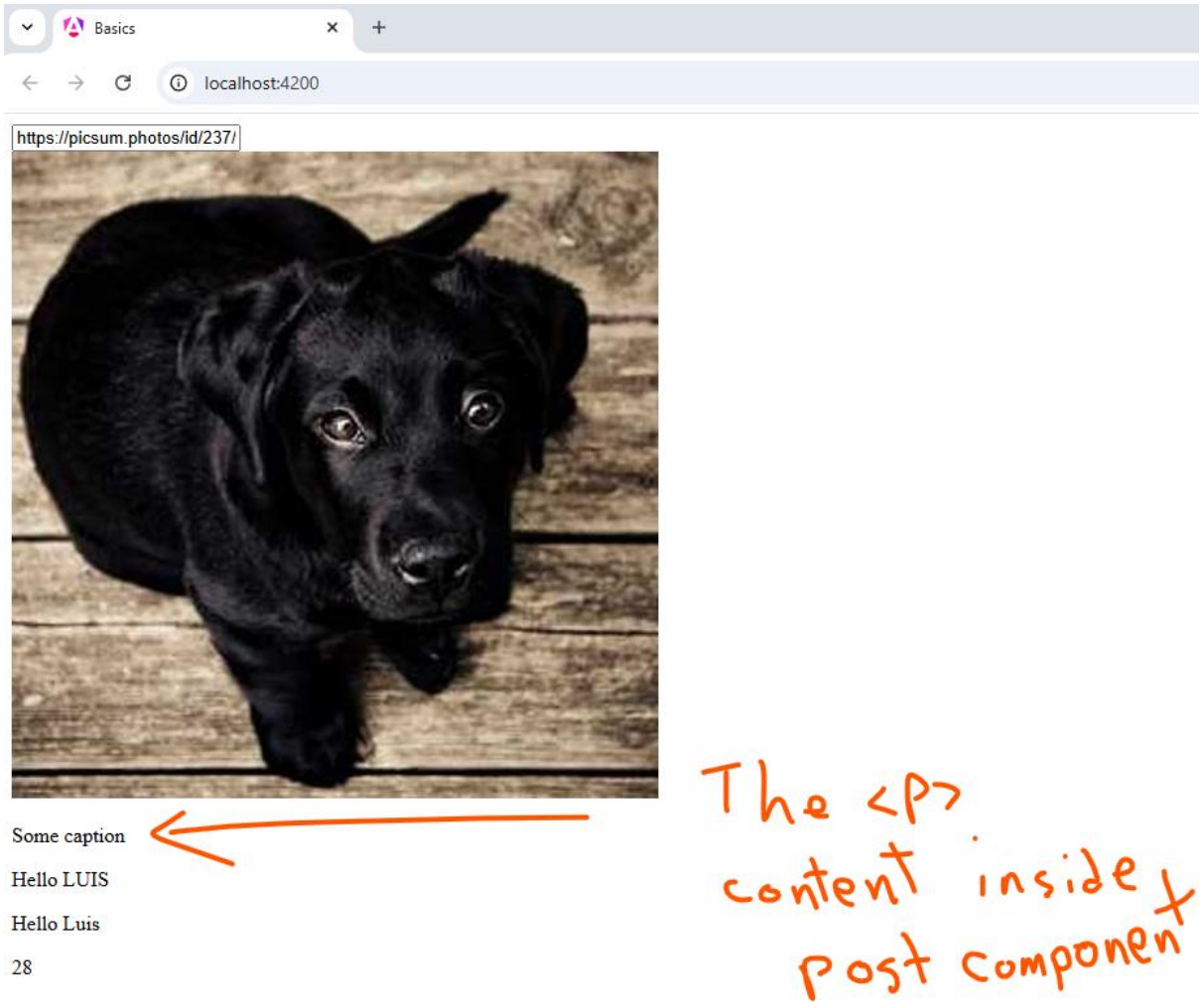
Angular creates a **custom element** for handling most of the work.

Below the img tag, we will add the **ng-content** elements.

The **<ng-content></ng-content>** element will search for **content inserted into our components tangs, if it finds something, the element will be replaced with the content. If nothing is found, the component won't render anything.**



Inserting content is completely optional. Even if we have the **ng-content** inside our templates, let's check out the browser:



The caption has been added to our template. By using **Content Projection**, we can pass down **HTML content from the parent component to the child component. It's another option at our disposal.**

Many libraries will use Content Projection for creating Skelton components. We can extend these libraries by inserting HTML content.