

Custom Properties

In this lecture, we're going to render our image inside the post components.

We have a couple of options at our disposal.

We can move the template and properties to the post component.

Alternatively, we can move just the template.

The URL for the image can be left inside the app component. Instead, we can feed the post component the image, why would we do this?

Our goal should be to develop reusable components. For example, let's say we're developing a blog, if our blog has hundreds of posts, would you rather create a component for every post or a single component?

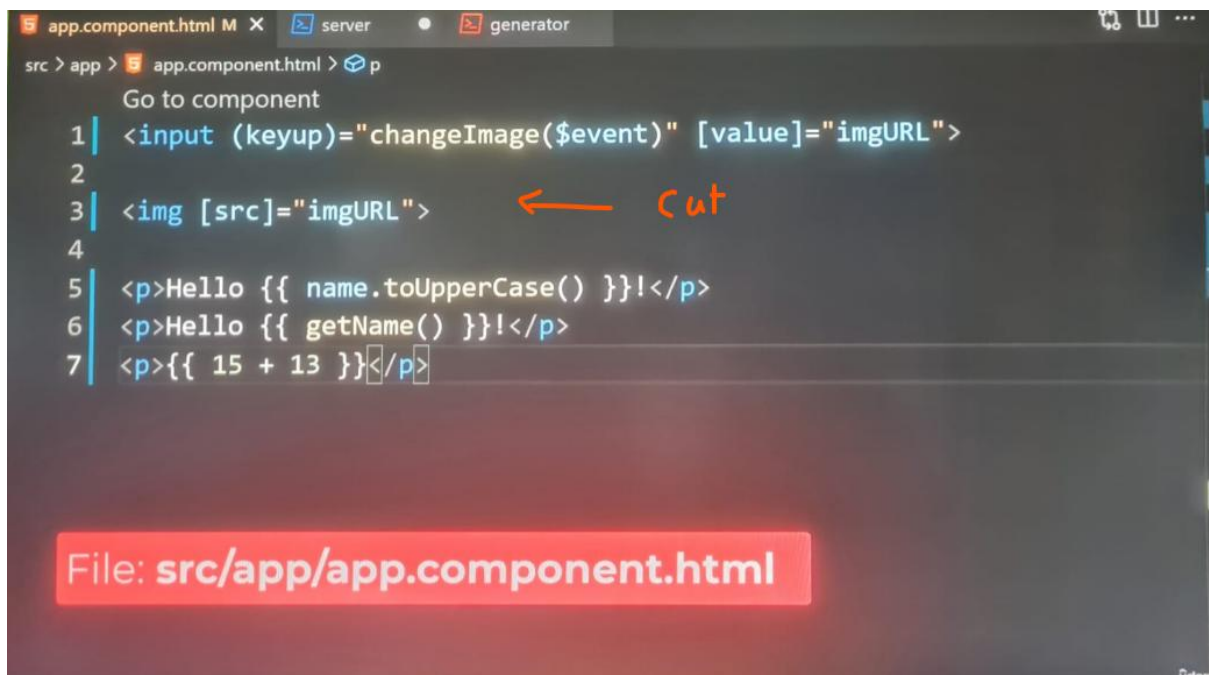
I bet you would want to create a single component. It's much more efficient instead of hard coding data into a component, we can create components to **accept data from a parent component**.

The post component can allow the parent component to tell it what image to render by sending the image to the component.

We can reuse the post component with different images. The responsibility of supplying the image will be the job of the parent component. If we hard code the image inside the post component, the same image will get rendered.

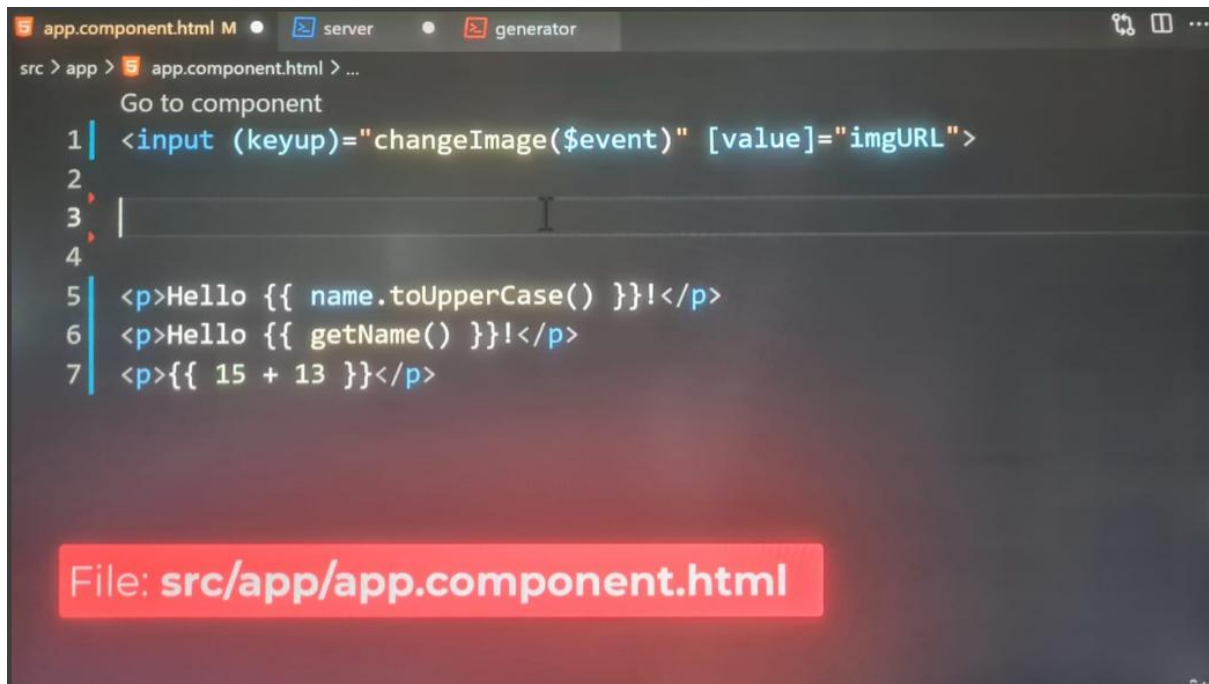
Angular makes it super easy to pass down data from component to component.

Let's get started.



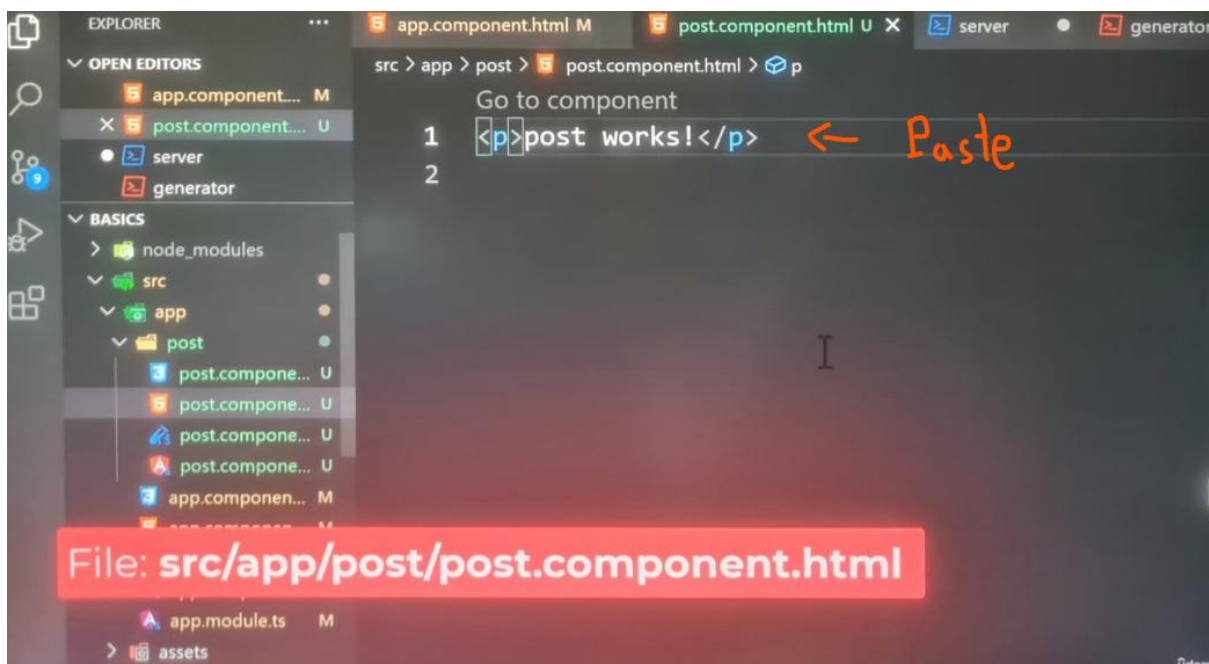
```
src > app > app.component.html > p
Go to component
1 | <input (keyup)="changeImage($event)" [value]="imgURL">
2 |
3 | <img [src]="imgURL"> ← cut
4 |
5 | <p>Hello {{ name.toUpperCase() }}!</p>
6 | <p>Hello {{ getName() }}!</p>
7 | <p>{{ 15 + 13 }}</p>
```

File: src/app/app.component.html



```
src > app > app.component.html > ...
Go to component
1 | <input (keyup)="changeImage($event)" [value]="imgURL">
2 |
3 |
4 |
5 | <p>Hello {{ name.toUpperCase() }}!</p>
6 | <p>Hello {{ getName() }}!</p>
7 | <p>{{ 15 + 13 }}</p>
```

File: **src/app/app.component.html**



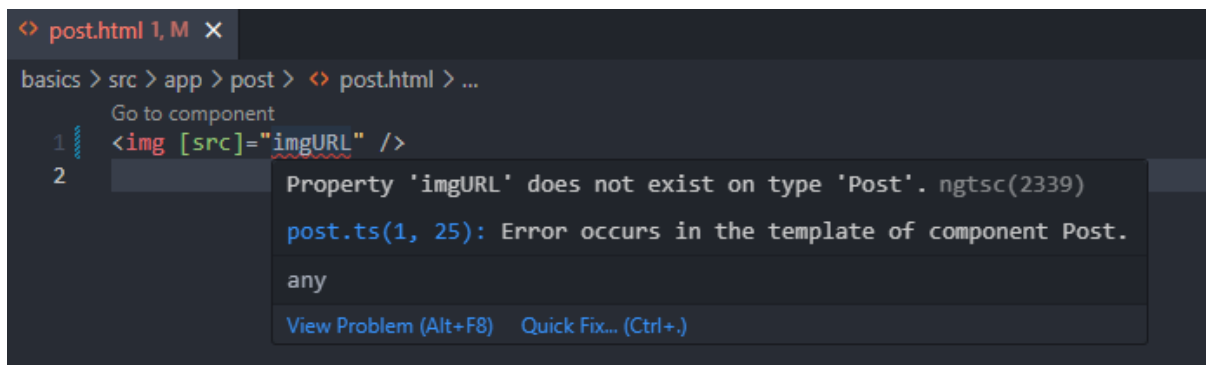
```
src > app > post > post.component.html > p
Go to component
1 | <p>post works!</p>
2 |
```

File: **src/app/post/post.component.html**



The editor will throw an error, it tells us the `imgURL` property does not exist in the post component class.

This is important to understand, we don't have access to properties in other classes.



Templates have limited scope to the class they are associated with.

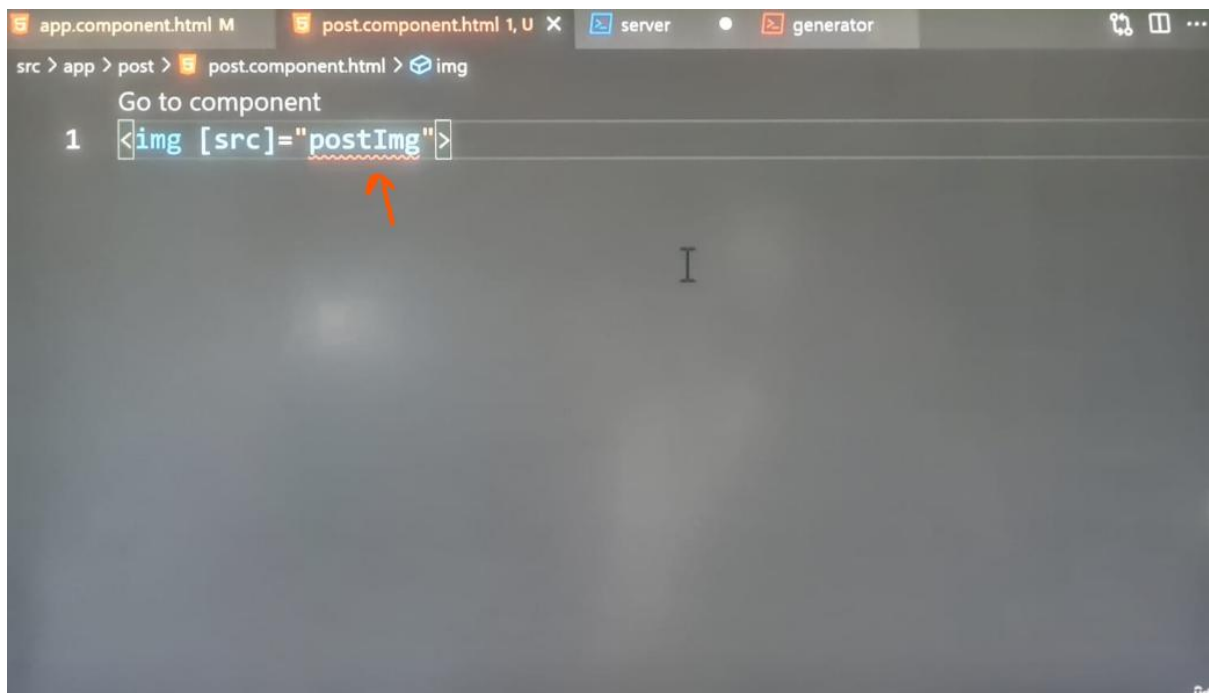
We should tell Angular **this property should come from the parent component.**

Before we move on, I want to rename the property to `postImg`.

The property will be called `postImg` in the **post component**.

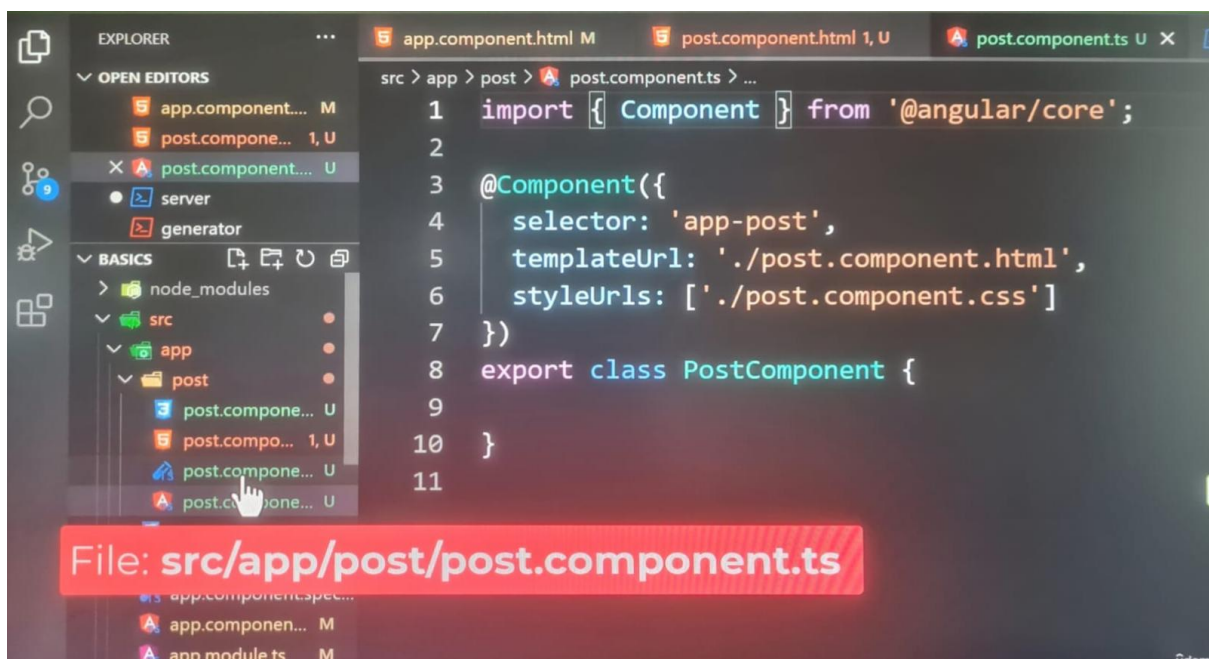
In the **app component**, the property will remain `imgURL`.

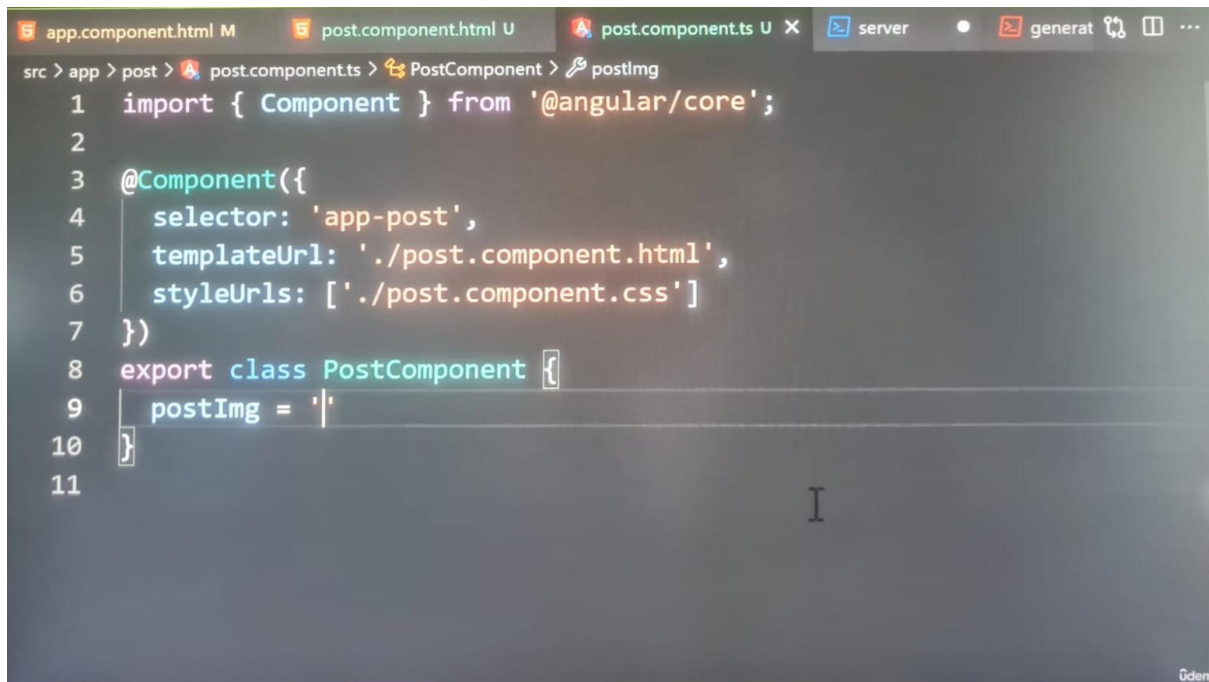
They will both hold the same value.



Inside post component class, we will add the postImage property with an initial value of an empty string, the property will be initialized with an empty string.

The goal is to update this property with the value **passed down by the parent component**:





```
src > app > post > post.component.ts > PostComponent > postImg
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-post',
5    templateUrl: './post.component.html',
6    styleUrls: ['./post.component.css']
7  })
8  export class PostComponent {
9    postImg = ''
10 }
11
```

This is where things get tricky.

Angular will prevent external code from changing this property. It can only be configured from within this class. That's a good thing.

We don't want to allow other classes to change their properties in this class without permission. It's the default behaviour of Angular.

However, in this specific case, we want to permit other components to set this property. We can tell Angular to permit parent components by adding a Decorator. At the import statement for the **@angular/core package**, we're going to update the list by adding a **Decorator** called **Input**.

The **Input function** is a Decorator.

Intrestingly, Decorators are not limited to classes, they can be applied to properties. The syntax is the exact same.

Decorators can be applied to properties by adding them before the property name.

Let's apply it to the postImage property:


```

src > app > post > post.component.ts > PostComponent > postImg
1  import { Component, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-post',
5    templateUrl: './post.component.html',
6    styleUrls: ['./post.component.css']
7  })
8  export class PostComponent {
9    @Input() postImg = ''
10 }
11

```

Decorator function (pointing to @Component)

Decorator (pointing to @Input)

We are calling the `@Input()` decorator as a function. We don't need to pass in configuration options to this decorator. After adding this decorator, we're ready for the last step. We need to add the post component to the app components template. Before we do, take special note of the **selector property** in the **@Component configuration settings**. The selector for this component is **app-post**. By default, Angular will prefix our component selector names with the word app. It's considered good practice to prefix selectors. This prefix prevents our custom components from clashing with default HTML tags.

Back in the app template, we are going to add the **app-post component**.

We can write components in other components as long as they're registered with the same module.

The app component isn't importing the post component. It doesn't have to. Components declared within the same module are accessible to one another. It's one of the many benefits of using Angular as module system. We can centralize our imports into one file instead of across several.

After adding this component: `<app-post></app-post>`, we can start passing on the image. Believe it or not, we can reuse a feature we learned in an earlier lecture, passing down data can be accomplished through **property binding**.

Properties can be configured through attributes on a component tag.

The attributes name must be the same as the property in the class accepting the data.

In this example, the name of the property is called **postImg**, the **postImg** attribute should be set to the **imgURL** property. By default, attribute values are not processed as expressions.

Luckily, Angular supports **property binding on components**.

Let's wrap this property with square brackets []. The attribute value will be set to the **imgURL property**.

```
<app-post [postImg]="imgURL"></app-post>
```

```

src > app > app.component.html > ...
Go to component
1 | <input (keyup)="changeImage($event)" [value]="imgURL">
2 |
3 | <app-post [postImg]="imgURL"></app-post>
4 |
5 | <p>Hello {{ name.toUpperCase() }}!</p>
6 | <p>Hello {{ getName() }}!</p>
7 | <p>{{ 15 + 13 }}</p>

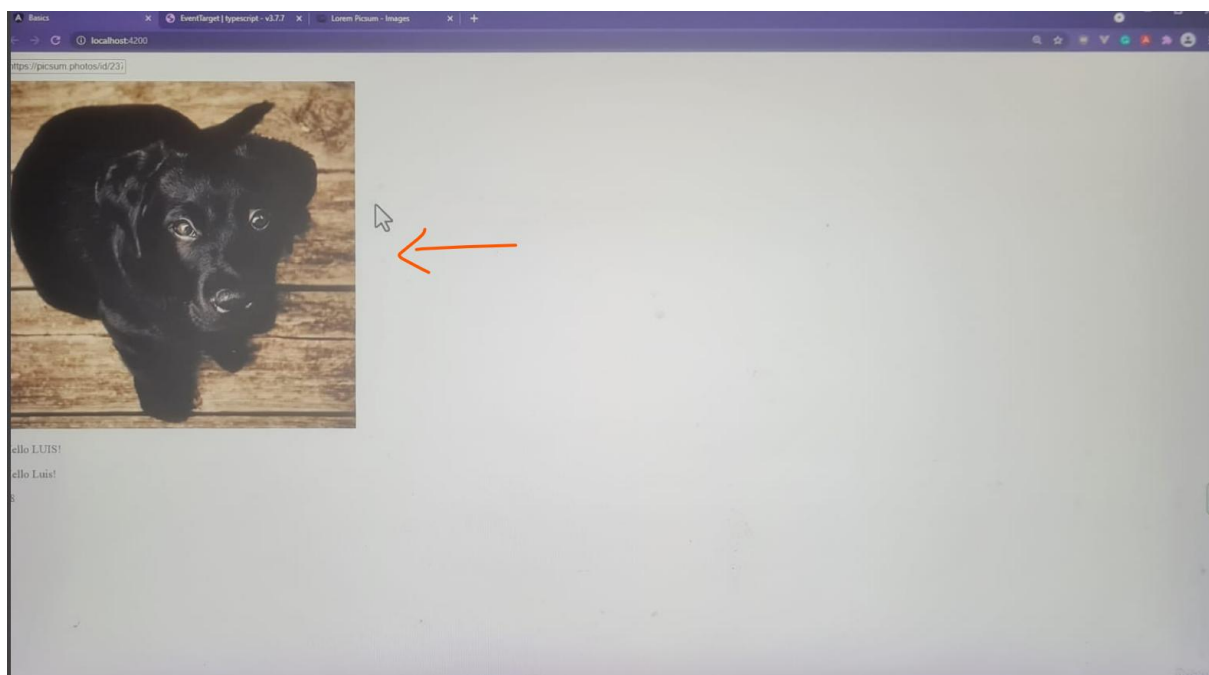
```

Handwritten annotations in orange:

- child component**: Points to the `<app-post>` tag.
- child attribute**: Points to the `[postImg]` attribute.
- Data Binding**: Points to the `imgURL` value in the `[value]` attribute of the `<input>` tag.
- Data passed from parent attribute**: Points to the `imgURL` value in the `[postImg]` attribute of the `<app-post>` tag.

The image is appearing on the page → We were able to communicate data between components, which is necessary for any app.

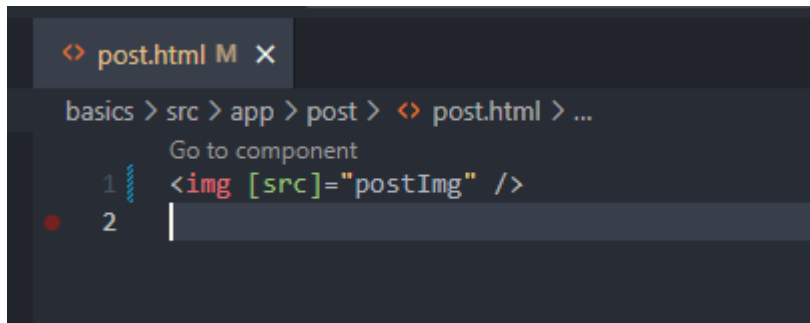
Data can flow from a parent component to a child components.



We will continue learning about data flow in components in the next set of lectures.

On my PC with latest Angular version of 20.0.5 with standalone architecture:

In post.html:

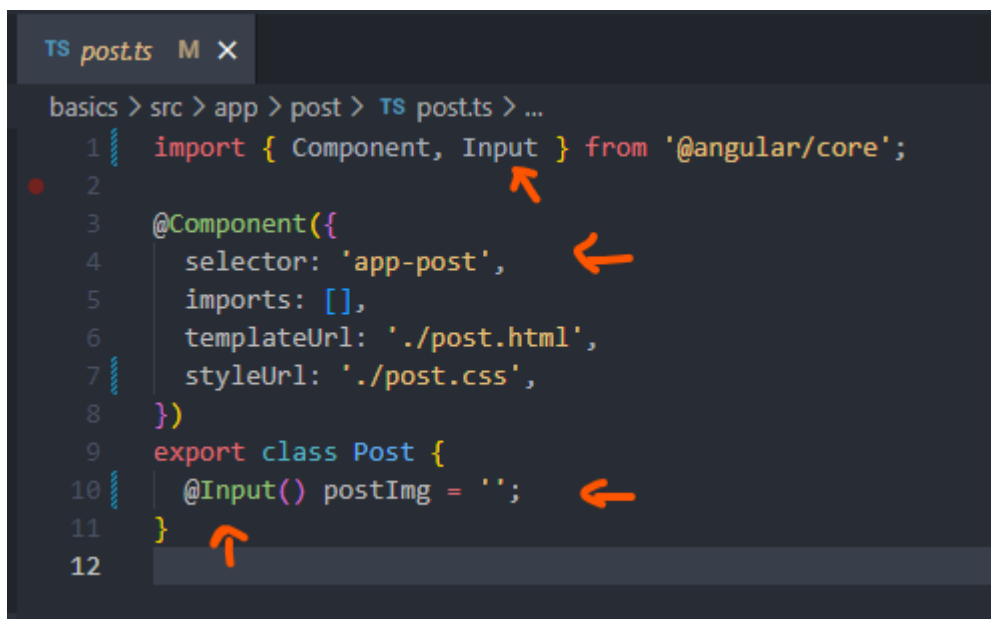


```

post.html M X
basics > src > app > post > post.html > ...
Go to component
1 <img [src]="postImg" />
2

```

In post.ts:



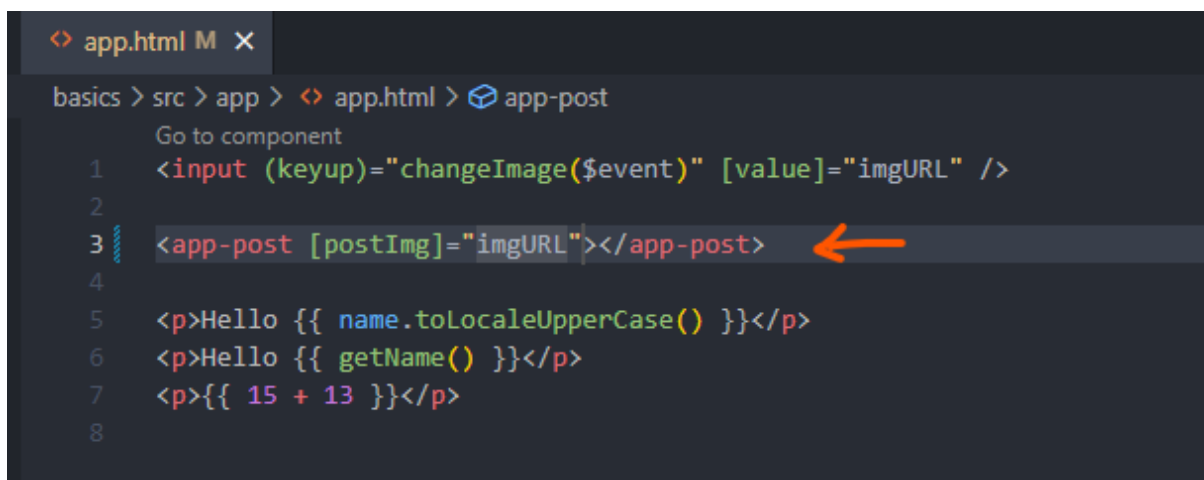
```

TS post.ts M X
basics > src > app > post > TS post.ts > ...
1 import { Component, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-post',
5   imports: [],
6   templateUrl: './post.html',
7   styleUrls: ['./post.css'],
8 })
9 export class Post {
10   @Input() postImg = '';
11 }
12

```

Annotations: An orange arrow points to the `Input` type in the import statement. Another orange arrow points to the `selector: 'app-post'` property in the `@Component` decorator. A third orange arrow points to the `@Input()` decorator in the `Post` class. A fourth orange arrow points to the closing brace of the `Post` class.

In app.ts:



```

app.html M X
basics > src > app > app.html > app-post
Go to component
1 <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3 <app-post [postImg]="imgURL"></app-post>
4
5 <p>Hello {{ name.toLocaleUpperCase() }}</p>
6 <p>Hello {{ getName() }}</p>
7 <p>{{ 15 + 13 }}</p>
8

```


Annotation: An orange arrow points to the `<app-post [postImg]="imgURL"></app-post>` line in the template.

In browser:

Basics x +

localhost:4200

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



←

Works

Hello LUIS

Hello Luis

28