

Interpolation

We can add expressions to our templates to output data



JavaScript Refresher: Expressions

An *expression* is a single line of code that evaluates to a value. The value may be a number, string or a logical value.



Expressions

```
// Number Expression
2 + 2; // Evaluates to 4

// String Expression
"hello".toUpperCase(); // Evaluates to HELLO WORLD

// Logical Expressions
100 === 100; // Evaluates to true
```



Not Expressions

```
// Declarations
var a;

// Function Declarations
function greet(message) {
  console.log(message);
}

// Conditional Statements
if (a === b) {
  // Do something
}
```



How Angular handles expressions

1. Searches for expressions
2. Runs the expression.
3. Replaces curly brackets with value from the expression.

{{ name }}

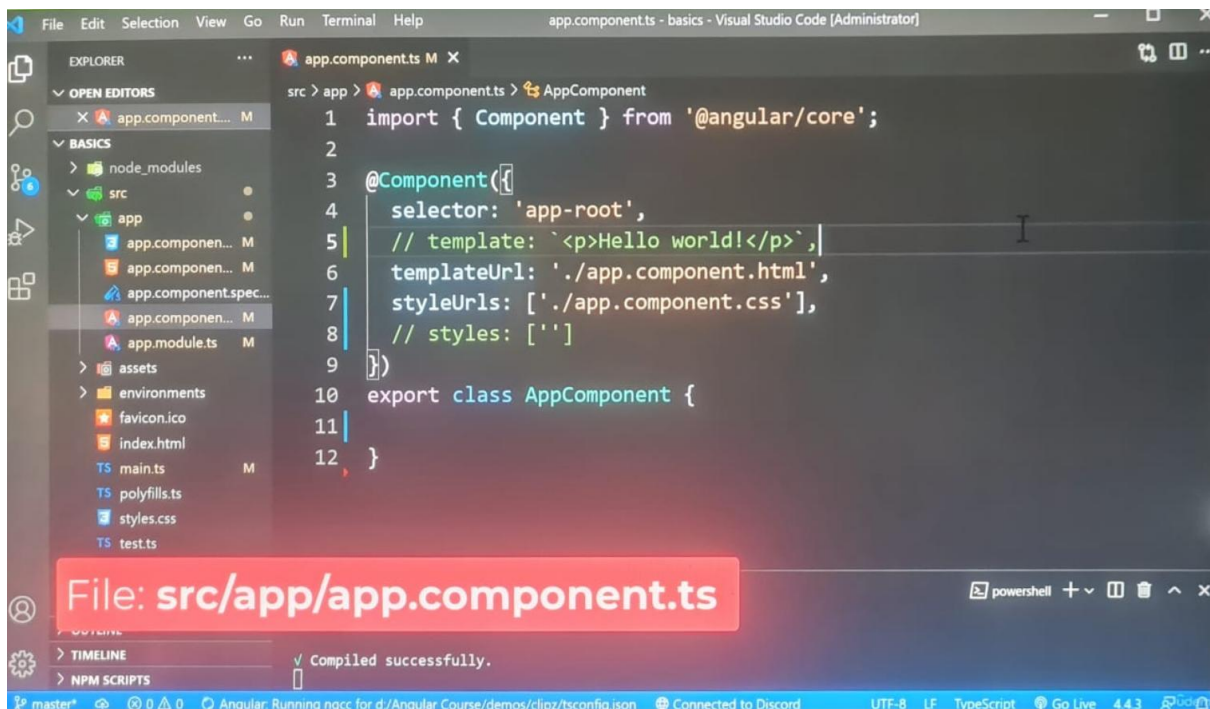


Difference between Expressions & String interpolation

Expressions are the code inside the curly brackets

String Interpolation is the process of replacing placeholders into string values

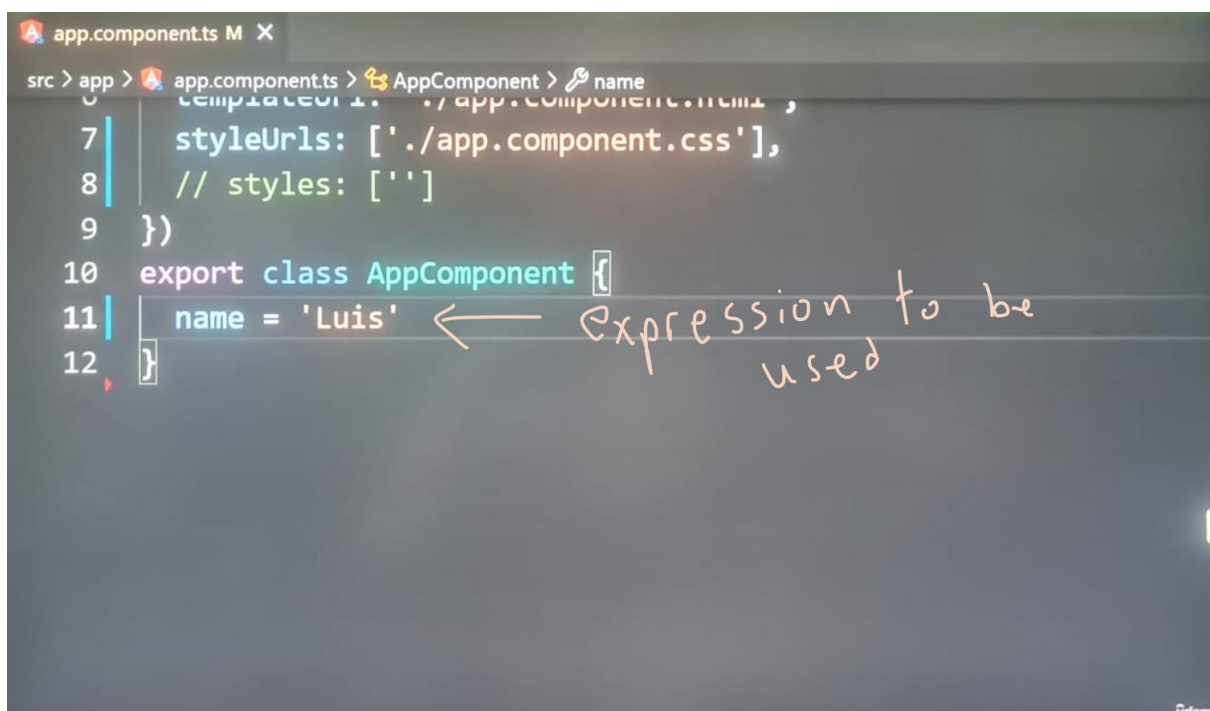
“The expression {{ name }} is interpolated into **John**.”



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure, including the 'src/app' directory. The main editor window shows the 'app.component.ts' file with the following code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   // template: `<p>Hello world!</p>`,
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css'],
8   // styles: ['']
9 })
10 export class AppComponent {
11
12 }
```

A red banner at the bottom of the editor window reads: "File: src/app/app.component.ts". The status bar at the bottom indicates "Compiled successfully."



This is a close-up view of the 'app.component.ts' file. The code is as follows:

```
7   styleUrls: ['./app.component.css'],
8   // styles: ['']
9 })
10 export class AppComponent {
11   name = 'Luis'
12 }
```

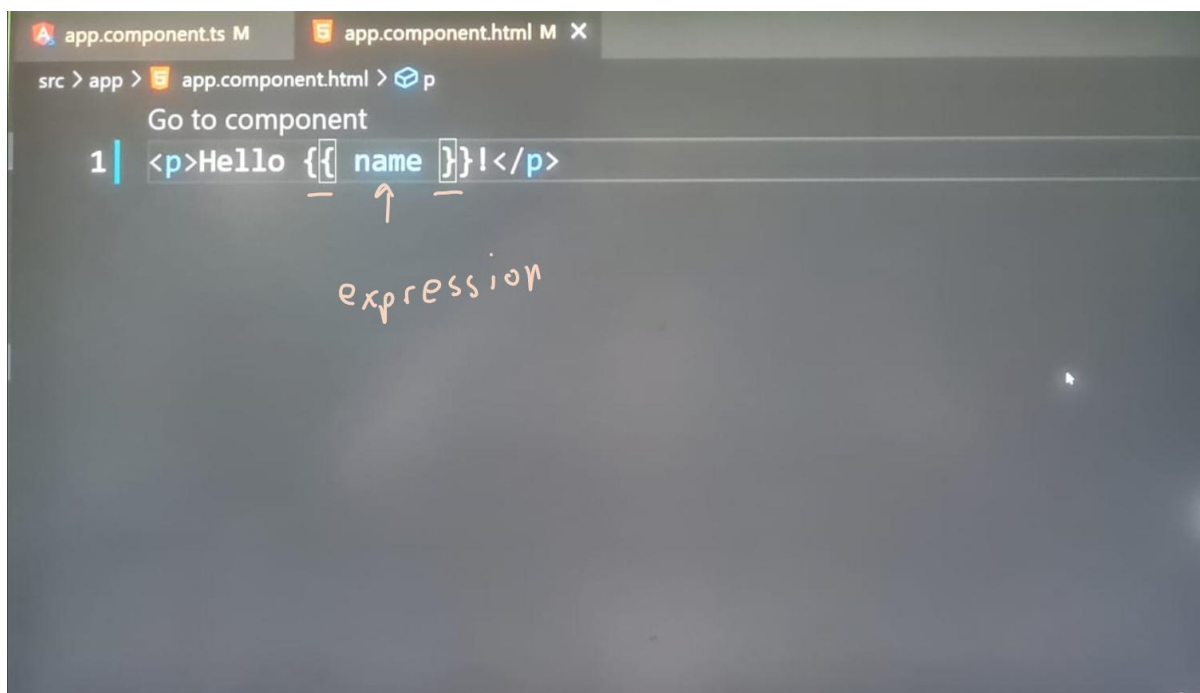
A handwritten note in orange ink, "Expression to be used", with an arrow pointing to the 'name = 'Luis'' line, is overlaid on the code.

```
app.component.ts M X
src > app > app.component.ts > AppComponent > name
1 | templateUrl: './app.component.html',
7 | styleUrls: ['./app.component.css'],
8 | // styles: []
9 | })
10 | export class AppComponent {
11 |   name = 'Luis'
12 | }
```

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

```
app.component.ts M app.component.html M X
src > app > app.component.html > p
Go to component
1 | <p>Hello world!</p> ← Update
I
```

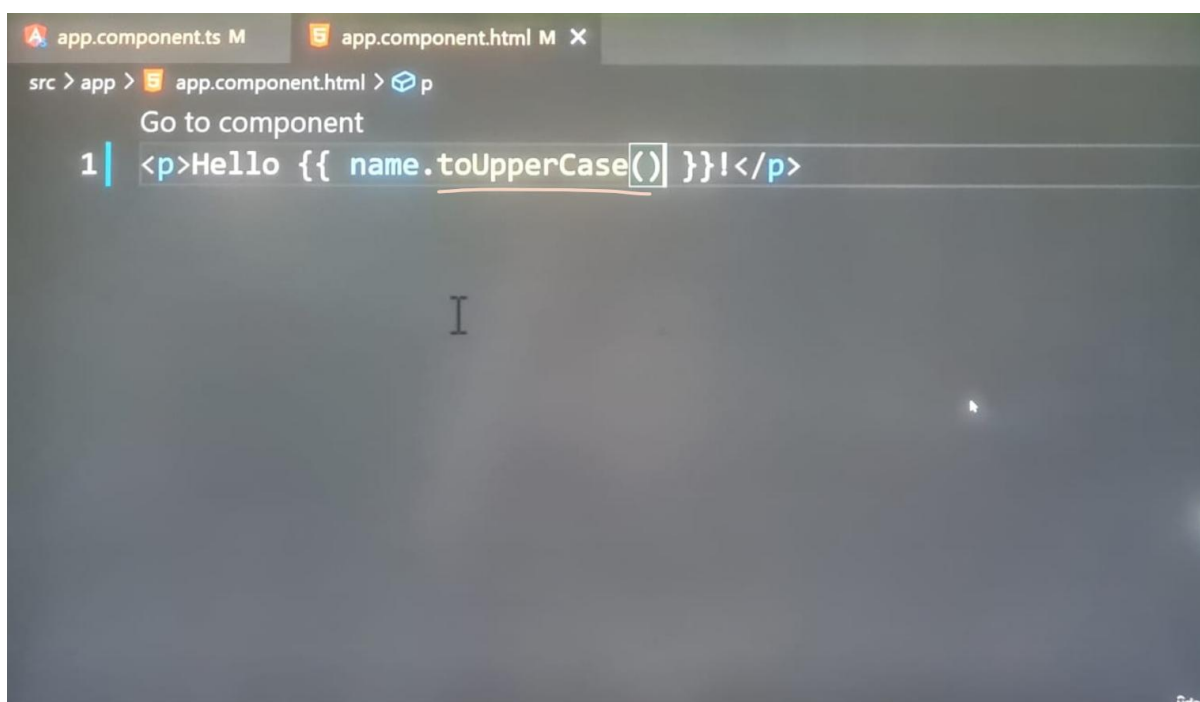
File: **src/app/app.component.html**



The screenshot shows an IDE with two tabs: 'app.component.ts' and 'app.component.html'. The 'app.component.html' tab is active, showing a file explorer path 'src > app > app.component.html' and a component selector 'p'. Below this, a message 'Go to component' is displayed. The main editor area contains a single line of code: `<p>Hello {{ name }}!</p>`. The text 'name' is highlighted with a blue selection box. A handwritten orange arrow points from the word 'expression' written below the code to the 'name' variable. A line number '1' is visible on the left side of the editor.

```
1 | <p>Hello {{ name }}!</p>
```

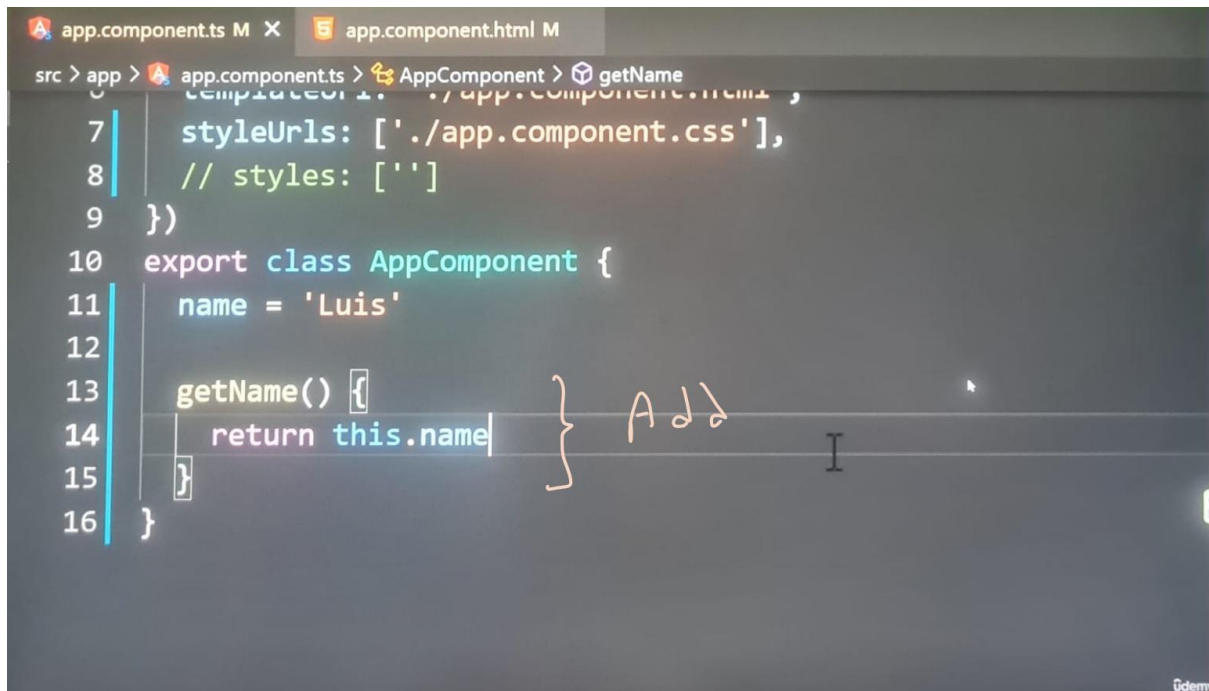
expression



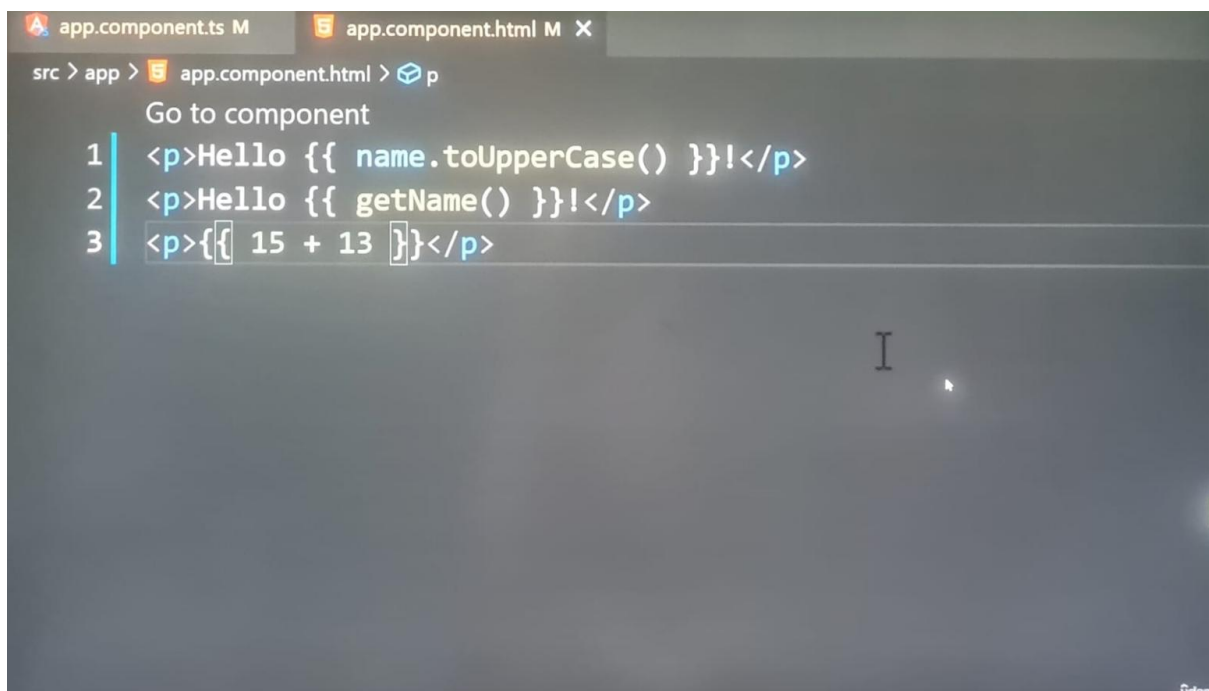
The screenshot shows the same IDE setup as the previous one. The code in the editor is now `<p>Hello {{ name.toUpperCase() }}!</p>`. The text 'name.toUpperCase()' is highlighted with a blue selection box. A handwritten orange underline is drawn under the entire expression 'name.toUpperCase()'. A large, bold, black letter 'I' is centered in the editor area below the code. The line number '1' remains on the left.

```
1 | <p>Hello {{ name.toUpperCase() }}!</p>
```

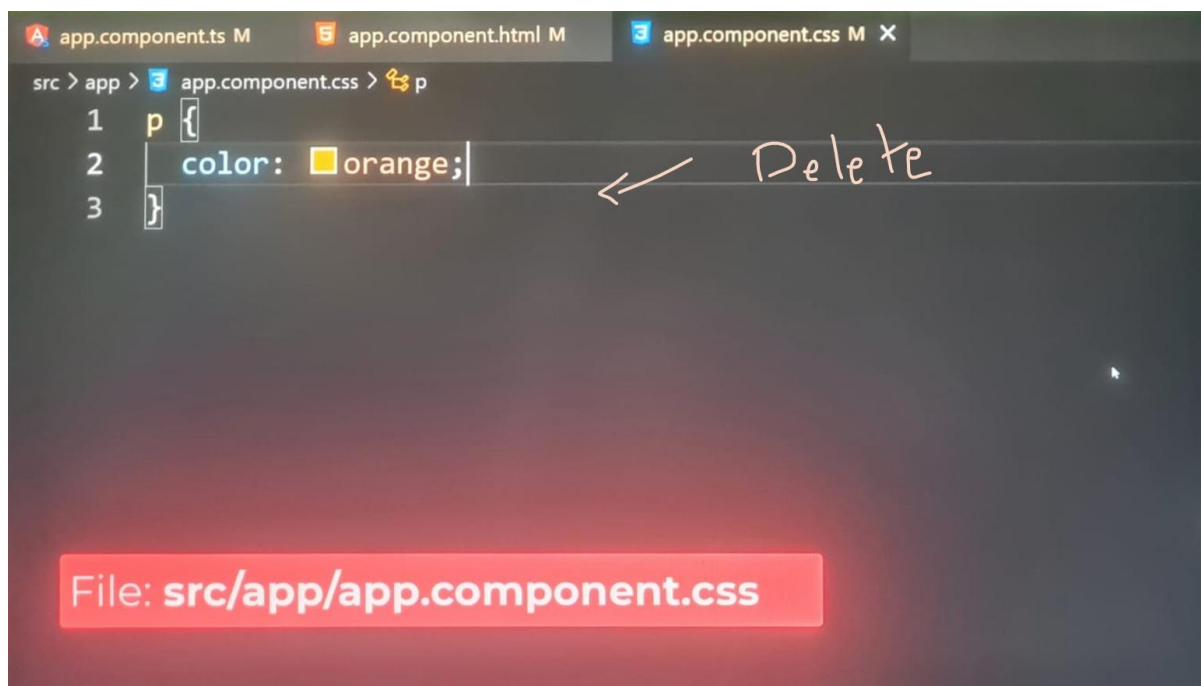
I



```
src > app > app.component.ts > AppComponent > getName
1  templateUrl: './app.component.html',
2  styleUrls: ['./app.component.css'],
3  // styles: ['']
4  })
5
6  export class AppComponent {
7    name = 'Luis'
8
9    getName() {
10     return this.name
11   }
12 }
13
14
15
16
```



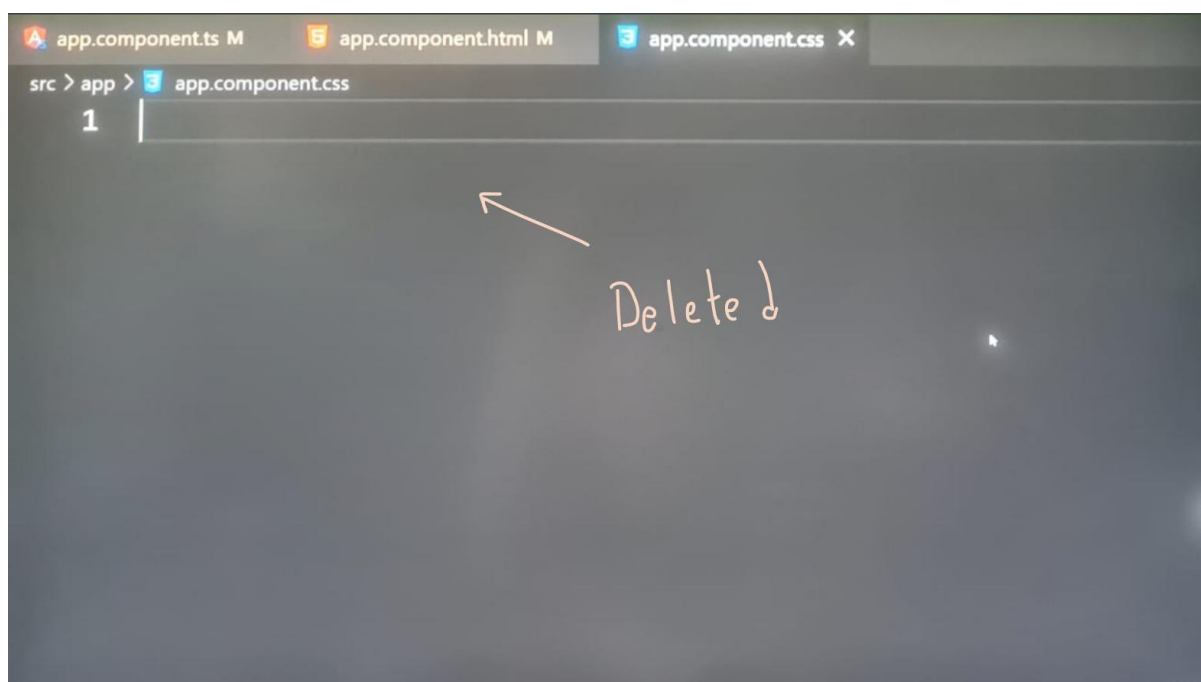
```
src > app > app.component.html > p
Go to component
1 <p>Hello {{ name.toUpperCase() }}!</p>
2 <p>Hello {{ getName() }}!</p>
3 <p>{{ 15 + 13 }}</p>
```

A screenshot of a code editor with three tabs: 'app.component.ts M', 'app.component.html M', and 'app.component.css M X'. The active tab is 'app.component.css M X'. The editor shows the following code:

```
src > app > app.component.css > p
1  p {
2    color: orange;
3  }
```

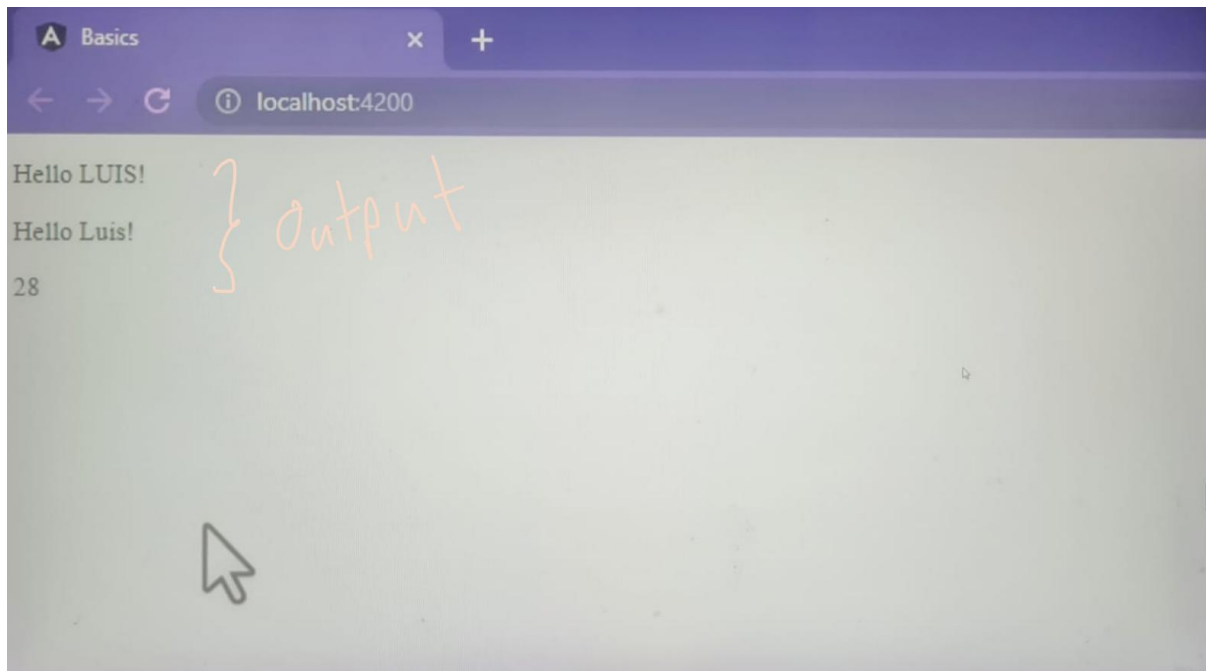
A handwritten arrow points to the line 'color: orange;' with the word 'Delete' written next to it. A red banner at the bottom of the editor displays the file path: 'File: src/app/app.component.css'.



A screenshot of the same code editor with the same three tabs. The active tab is 'app.component.css X'. The editor shows a single line with a cursor at the beginning:

```
src > app > app.component.css
1  |
```

A handwritten arrow points to the line with the word 'Delete' written next to it, indicating the file has been emptied.

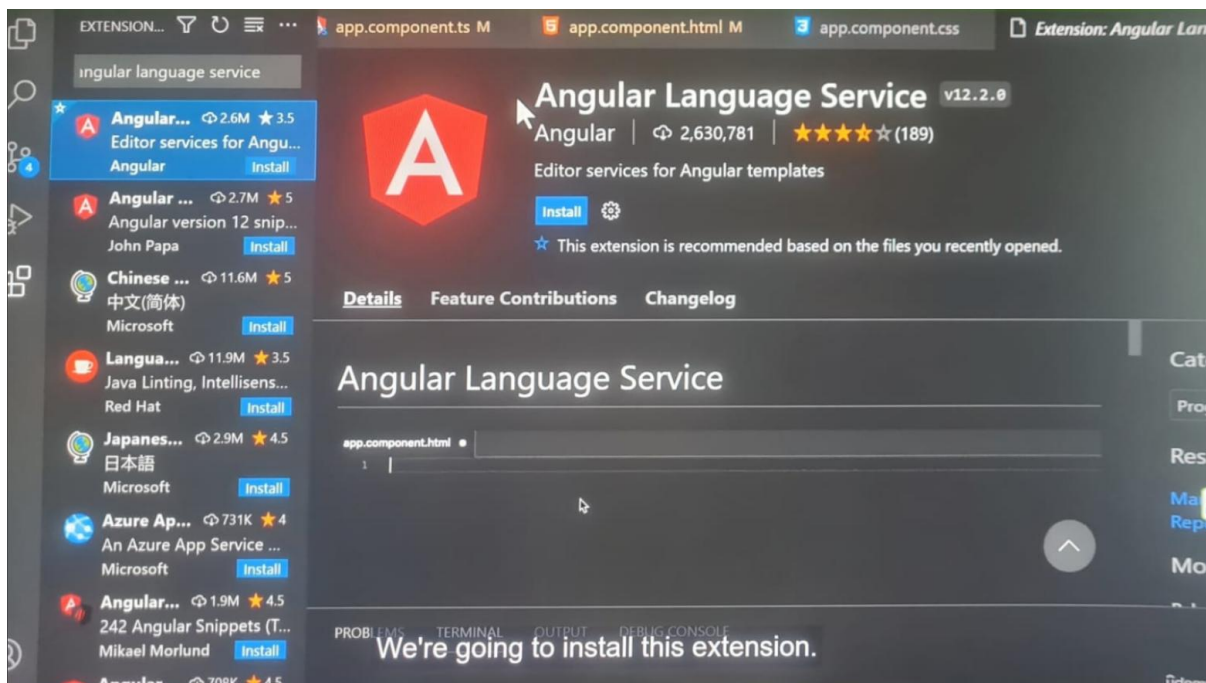


To recap, Interpolation is the process of replacing placeholders with string values, we can output properties by using double curly braces in our templates.

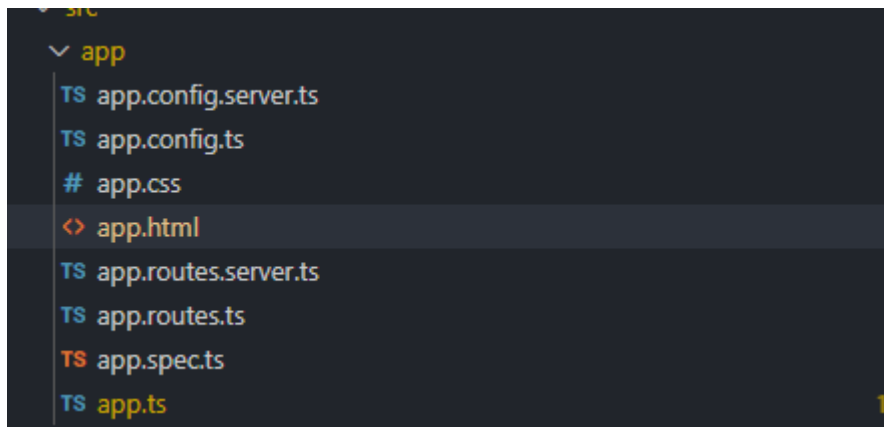
The code inside the curly braces must be an expression.

The evaluated value from the expression is what will get outputted.

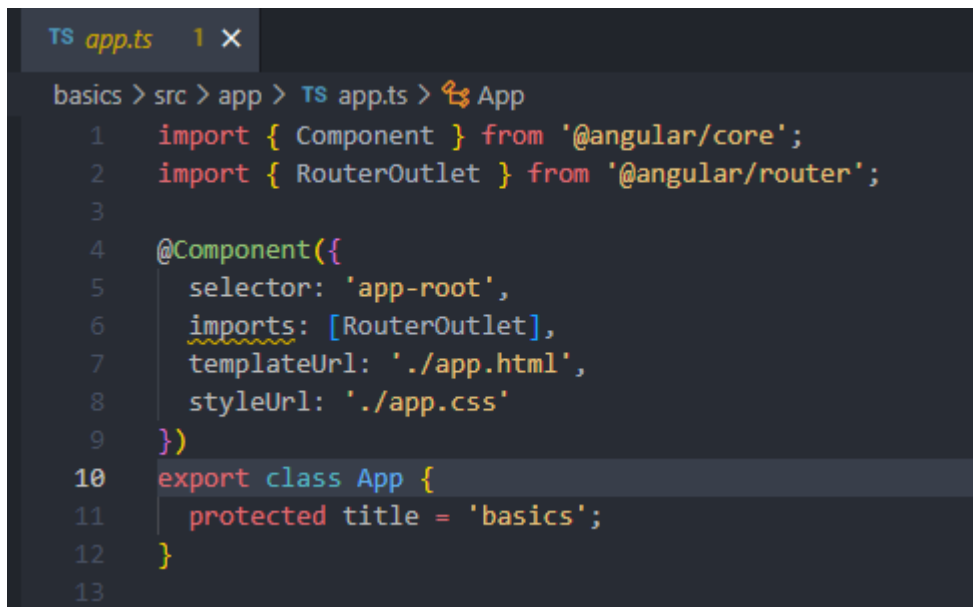
Angular will replace the curly braces with the evaluated value.



On my PC with the latest Angular Version:



app.ts == app.component.ts:



This is what we did in the old version:

```

src > app > app.component.ts > AppComponent > getName
1  templateUrl: './app.component.html',
7  styleUrls: ['./app.component.css'],
8  // styles: ['']
9  })
10 export class AppComponent {
11     name = 'Luis'
12
13     getName() {
14         return this.name
15     }
16 }

```

Handwritten annotations: "Add" and a closing brace for the `getName` method.

Now I will try to replicate that changes:

```

TS app.ts 1, M X
basics > src > app > TS app.ts > ...
1  import { Component } from '@angular/core';
2  import { RouterOutlet } from '@angular/router';
3
4  @Component({
5      selector: 'app-root',
6      imports: [RouterOutlet],
7      templateUrl: './app.html',
8      styleUrls: ['./app.css'],
9  })
10 export class App {
11     protected title = 'basics';
12
13     protected name = 'Luis';
14
15     getName() {
16         return this.name;
17     }
18 }

```

Handwritten annotations: A closing brace for the `App` class.

app.html == app.component.html:

Old version of app.component.html:

```

src > app > app.component.html > p
Go to component
1 <p>Hello {{ name.toUpperCase() }}!</p>
2 <p>Hello {{ getName() }}!</p>
3 <p>{{ 15 + 13 }}</p>

```

My replication of the app.html:

```

basics > src > app > app.html > ...
Go to component
1 <p>Hello {{ name.toLocaleUpperCase() }}</p>
2 <p>Hello {{ getName() }}</p>
3 <p>{{ 15 + 13 }}</p>
4

```

Output:

