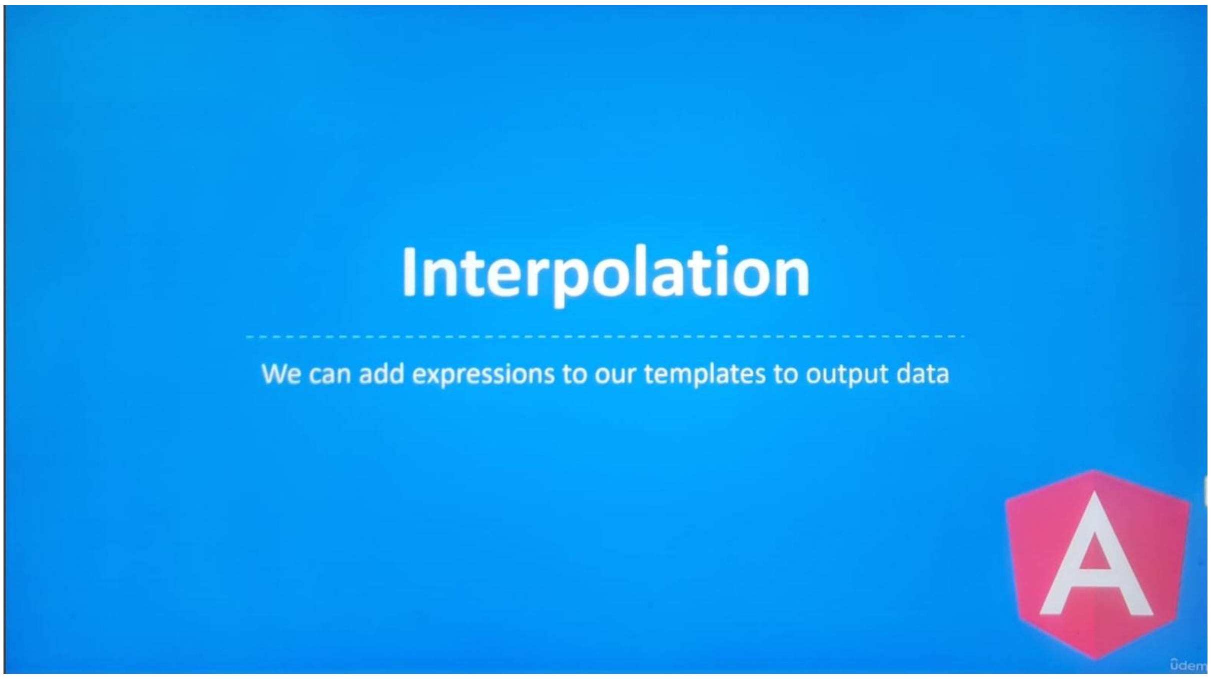
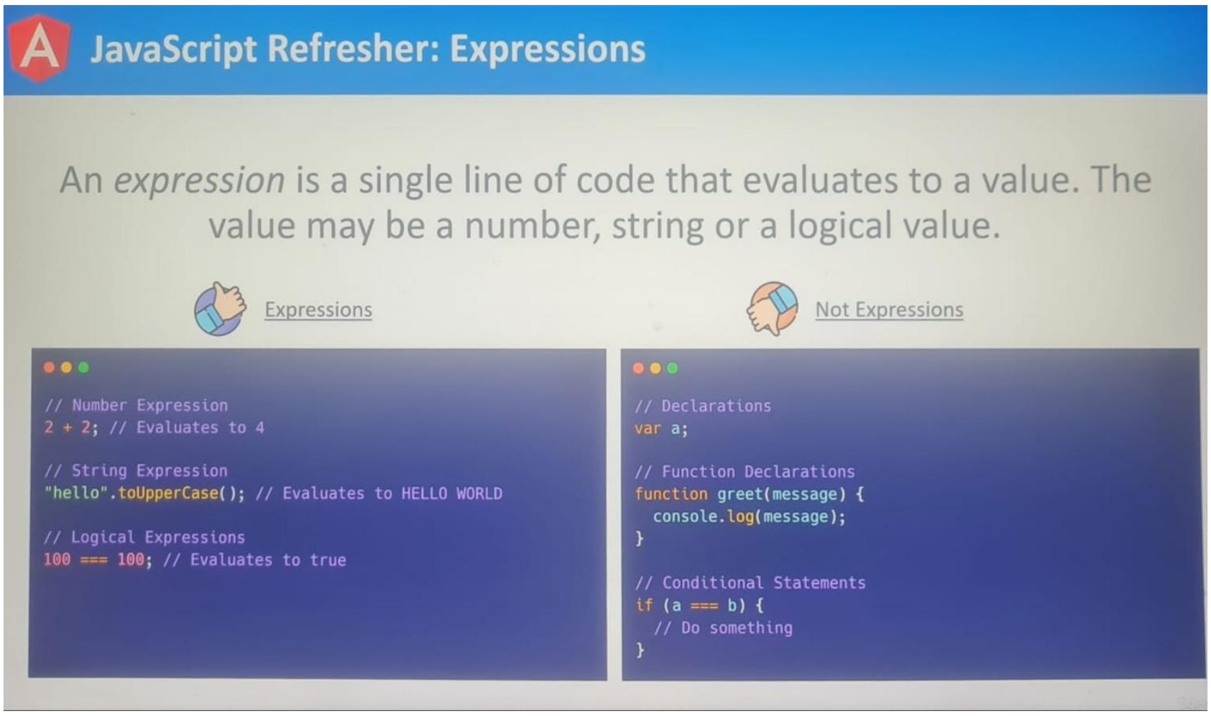


Interpolation

We can add expressions to our templates to output data



JavaScript Refresher: Expressions

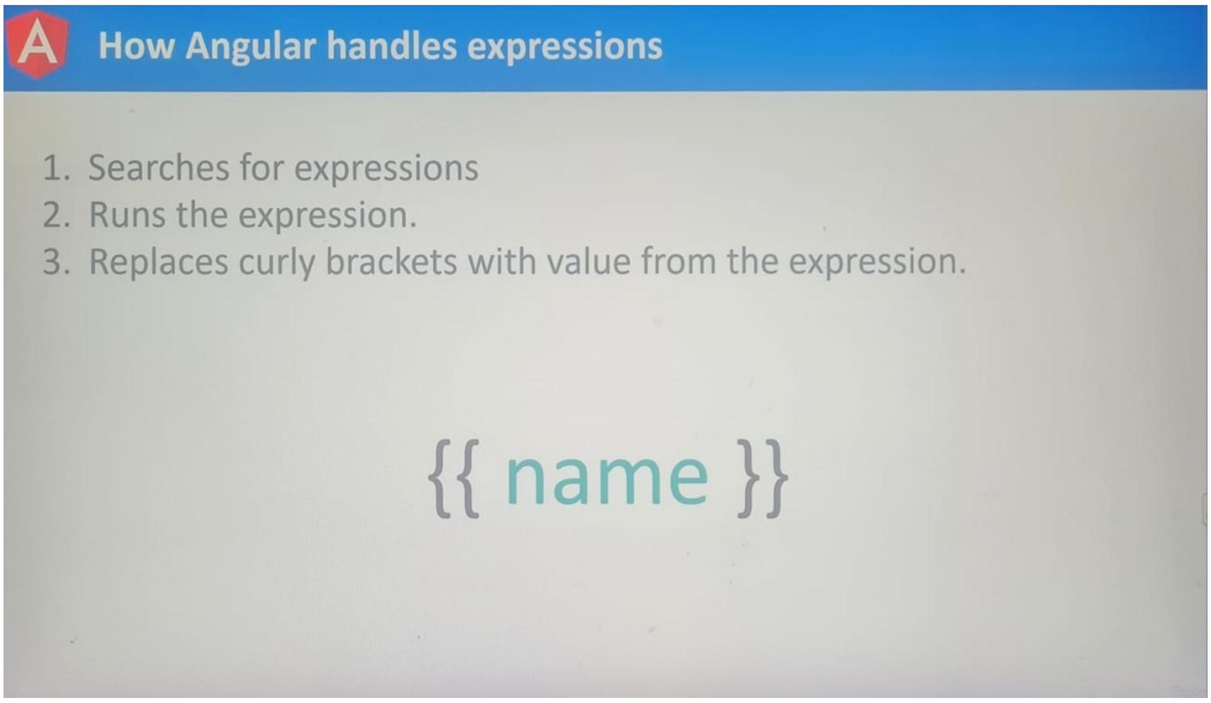
An *expression* is a single line of code that evaluates to a value. The value may be a number, string or a logical value.

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

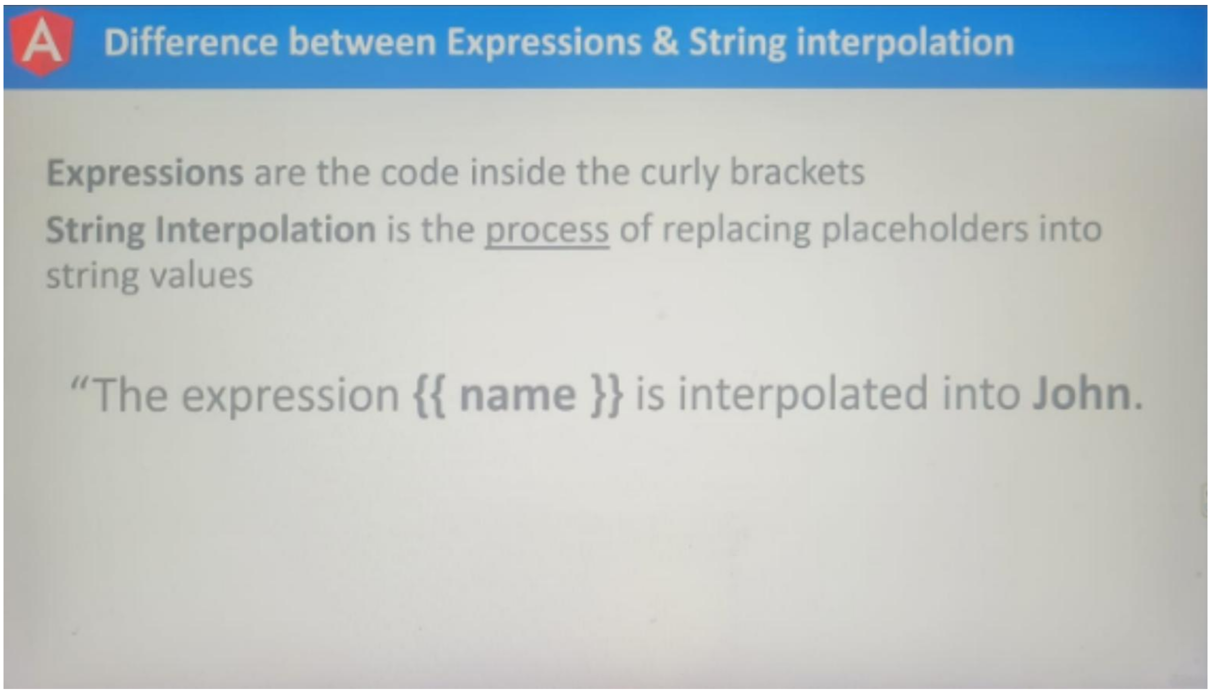
```
// Declarations
var a;

// 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 }}

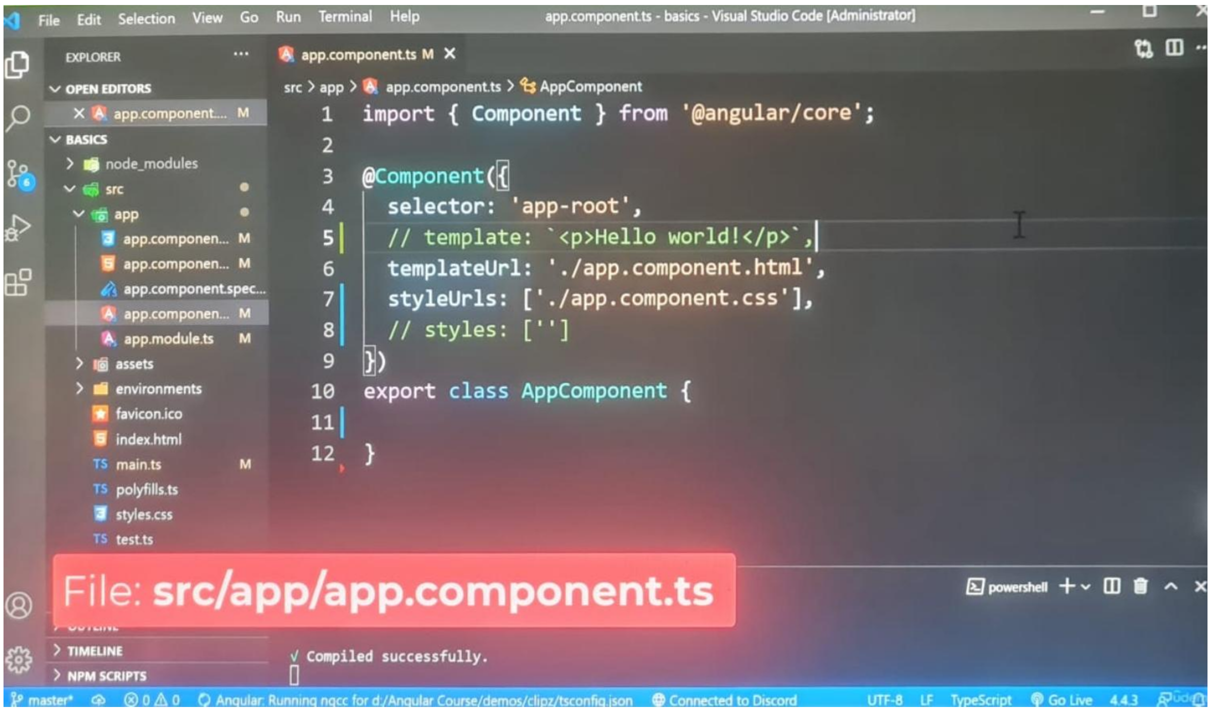


Difference between Expressions & String interpolation

Expressions are the code inside the curly brackets

String Interpolation is the process of replacing placeholders into string values

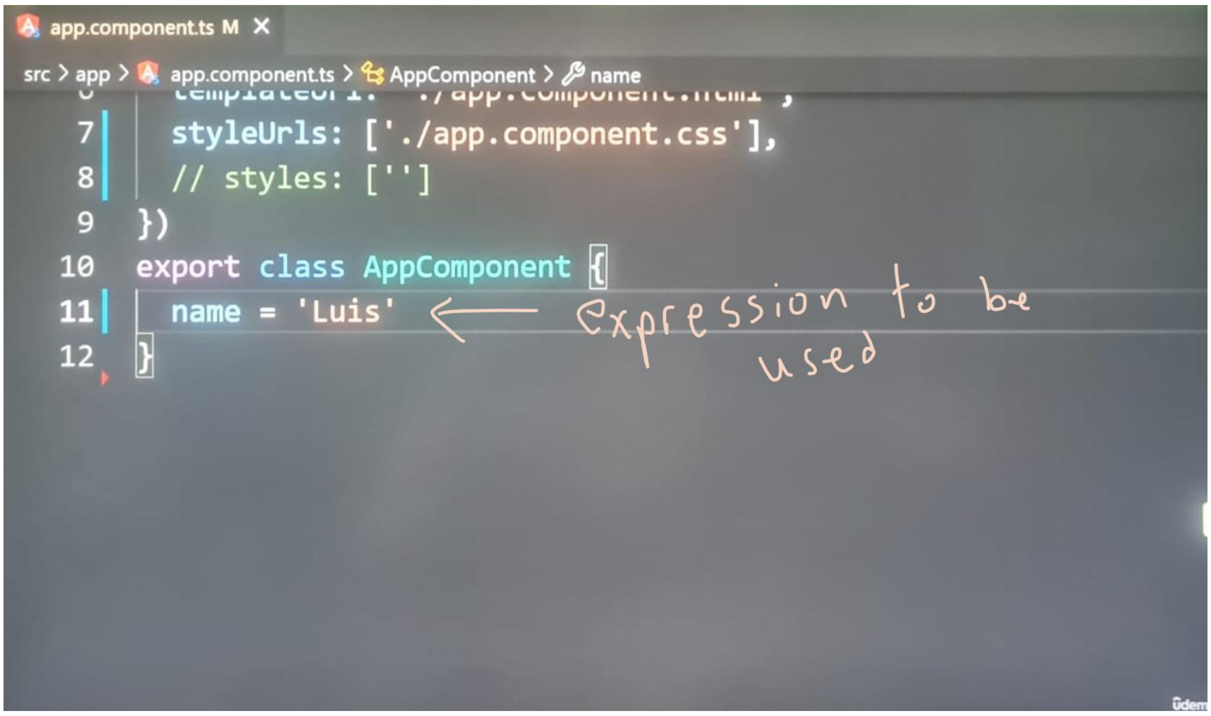
"The expression **{{ name }}** is interpolated into **John**.



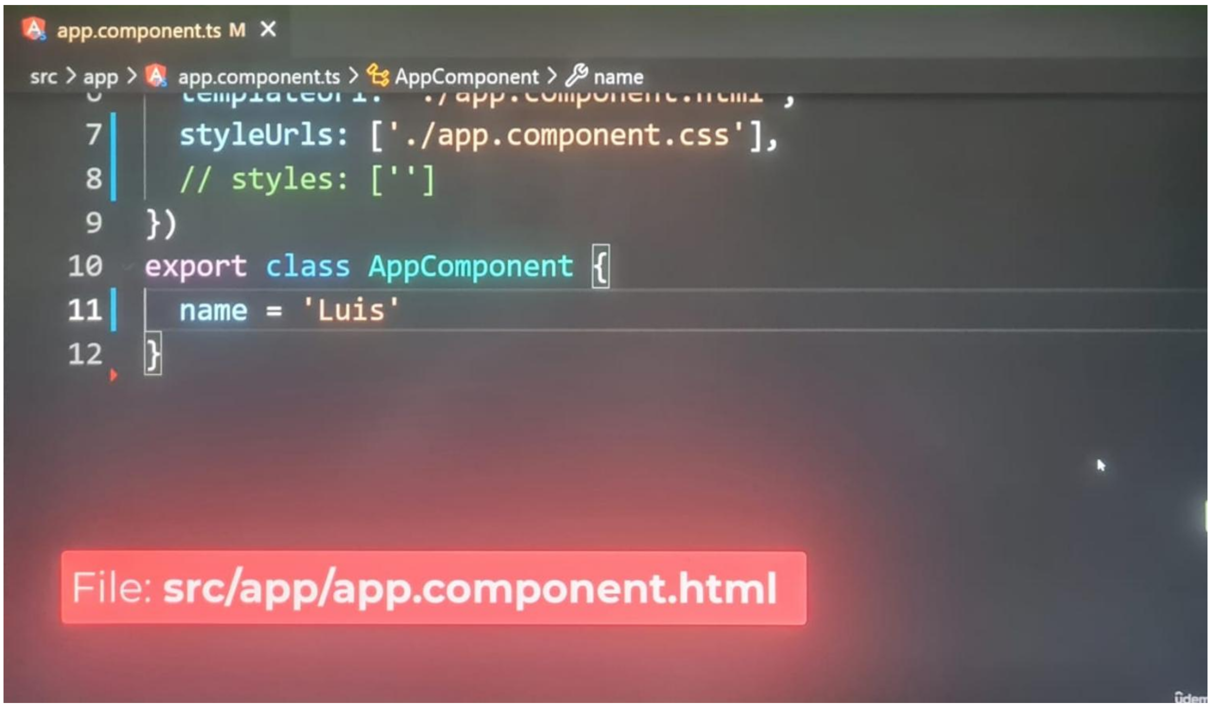
File: src/app/app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  // template: `<p>Hello world!</p>`,
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  // styles: ['']
})
export class AppComponent {

}
```

```
√ Compiled successfully.
```

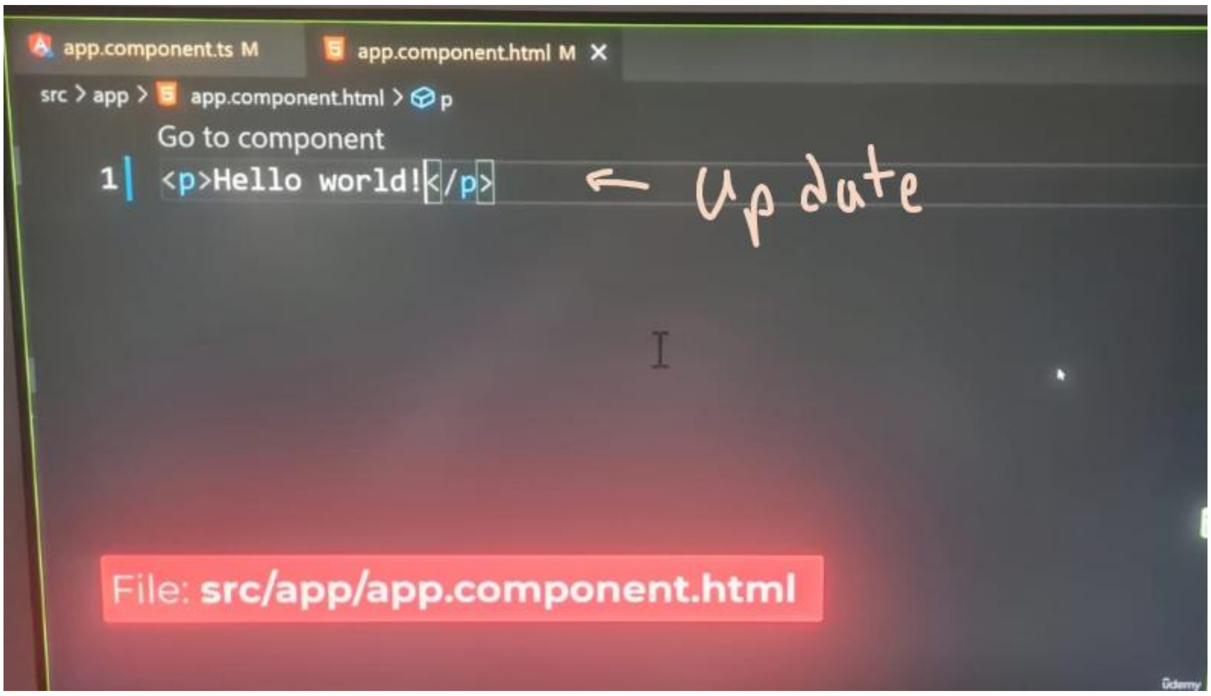


```
  styleUrls: ['./app.component.css'],
  // styles: ['']
})
export class AppComponent {
  name = 'Luis'
}
```



```
  styleUrls: ['./app.component.css'],
  // styles: ['']
})
export class AppComponent {
  name = 'Luis'
}
```

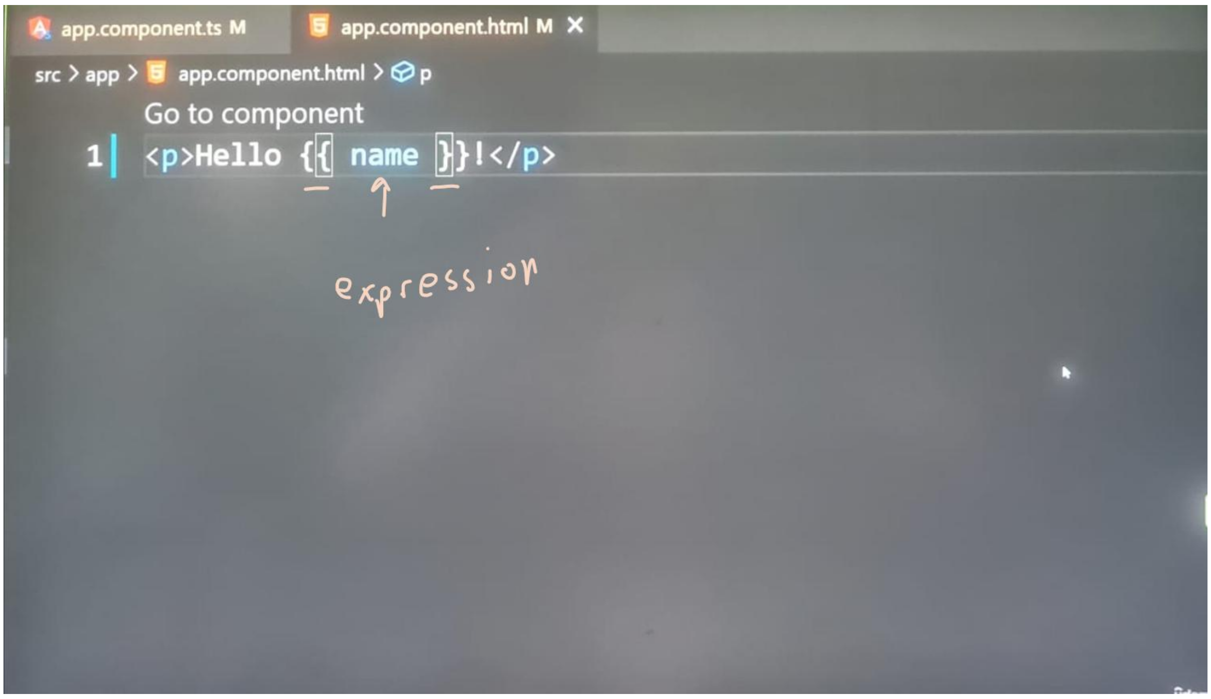
File: **src/app/app.component.html**



```
<p>Hello world!</p>
```

← Update

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

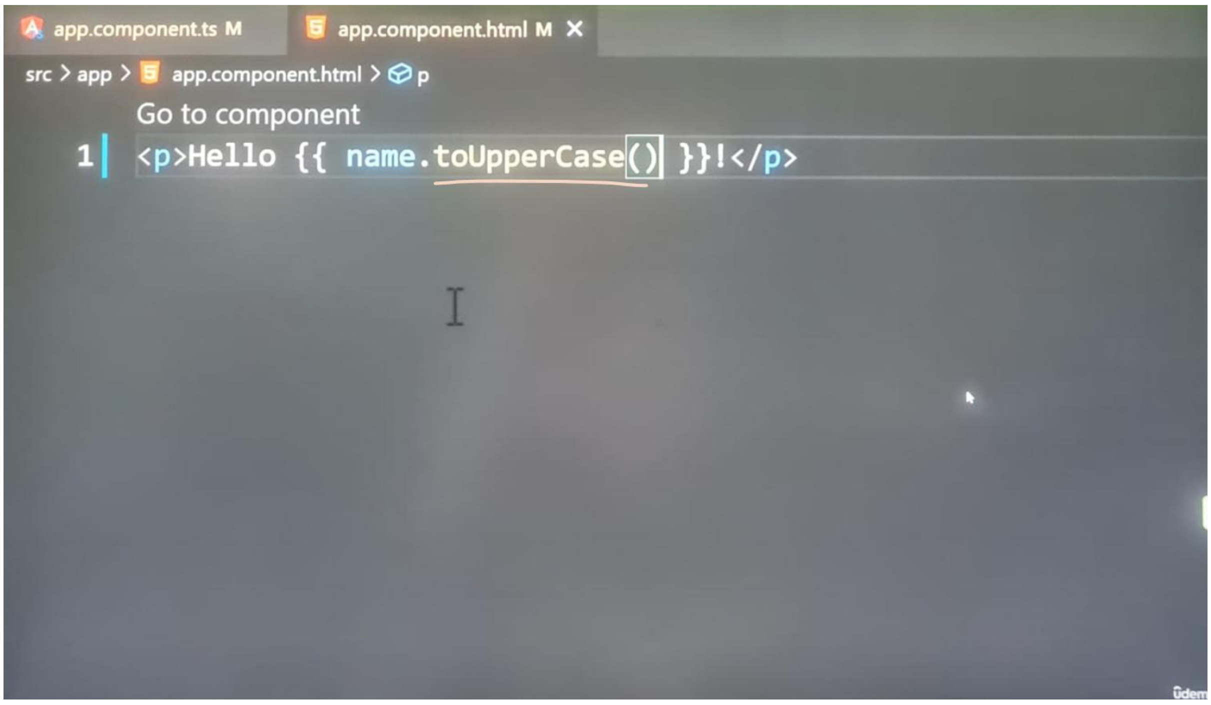


app.component.ts M | app.component.html M ×

src > app > app.component.html > p

Go to component

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

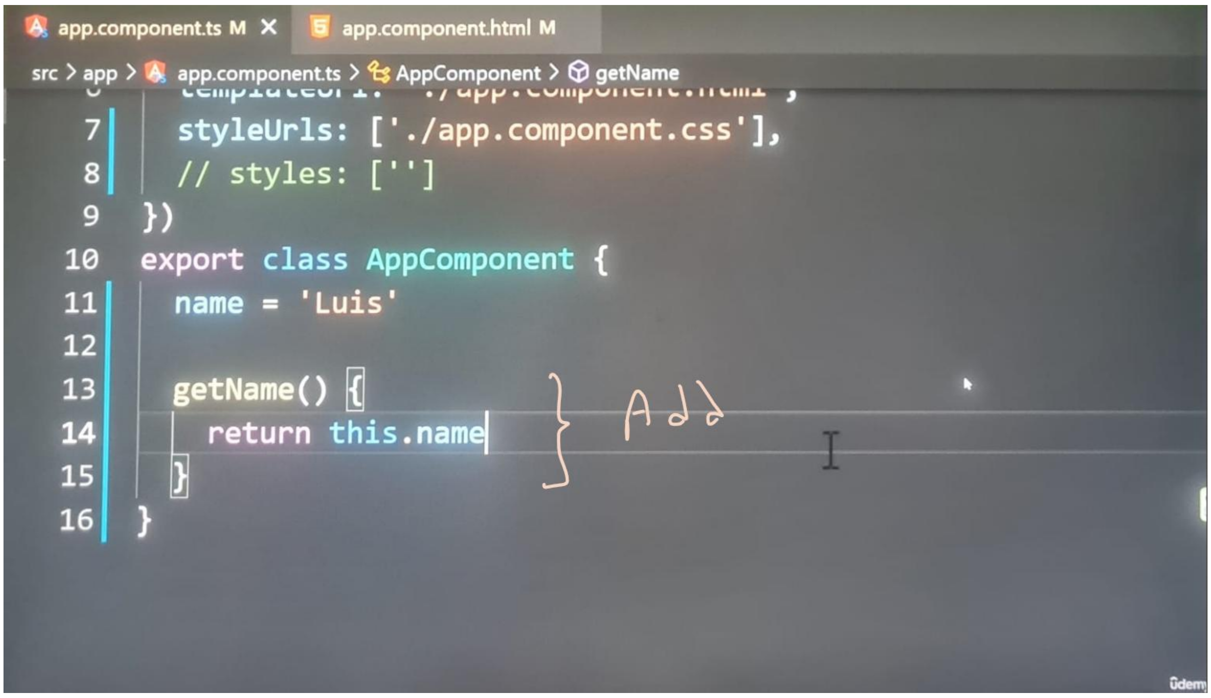


app.component.ts M | app.component.html M ×

src > app > app.component.html > p

Go to component

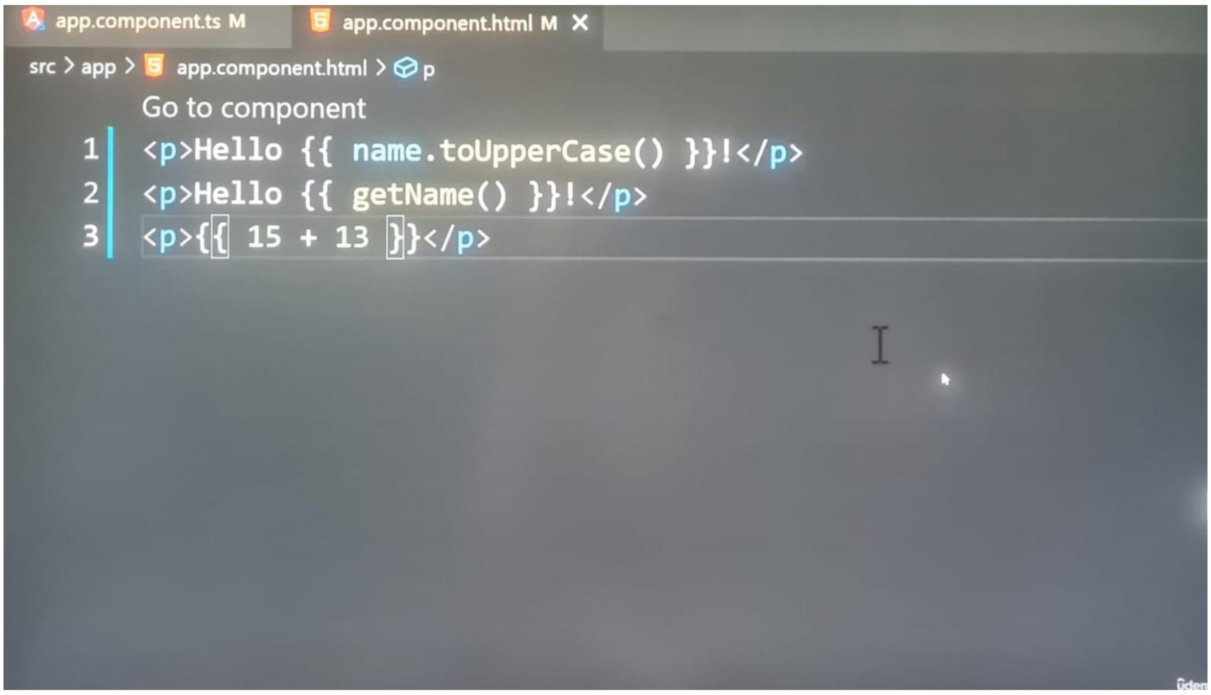
```
<p>Hello {{ name.toUpperCase() }}!</p>
```



app.component.ts M × app.component.html M

src > app > app.component.ts > AppComponent > getName

```
  styleUrls: ['./app.component.css'],
  // styles: ['']
})
export class AppComponent {
  name = 'Luis'

  getName() {
    return this.name
  }
}
```

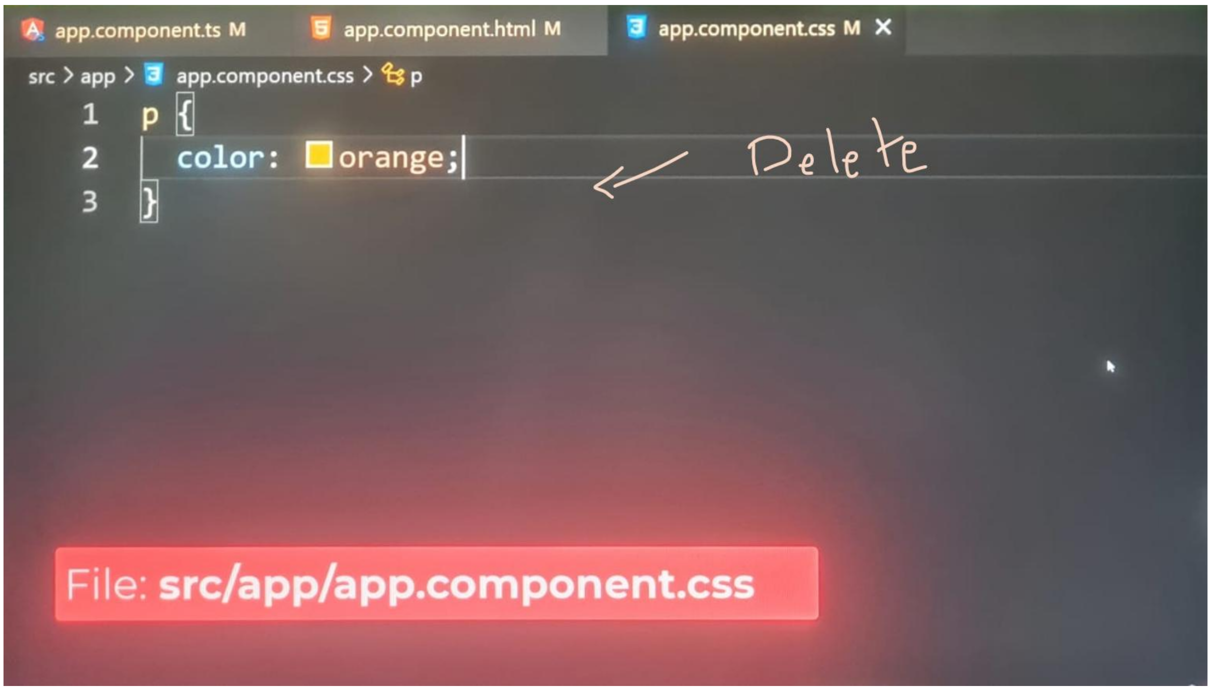


app.component.ts M app.component.html M ×

src > app > app.component.html > p

Go to component

```
<p>Hello {{ name.toUpperCase() }}!</p>
<p>Hello {{ getName() }}!</p>
<p>{{ 15 + 13 }}</p>
```



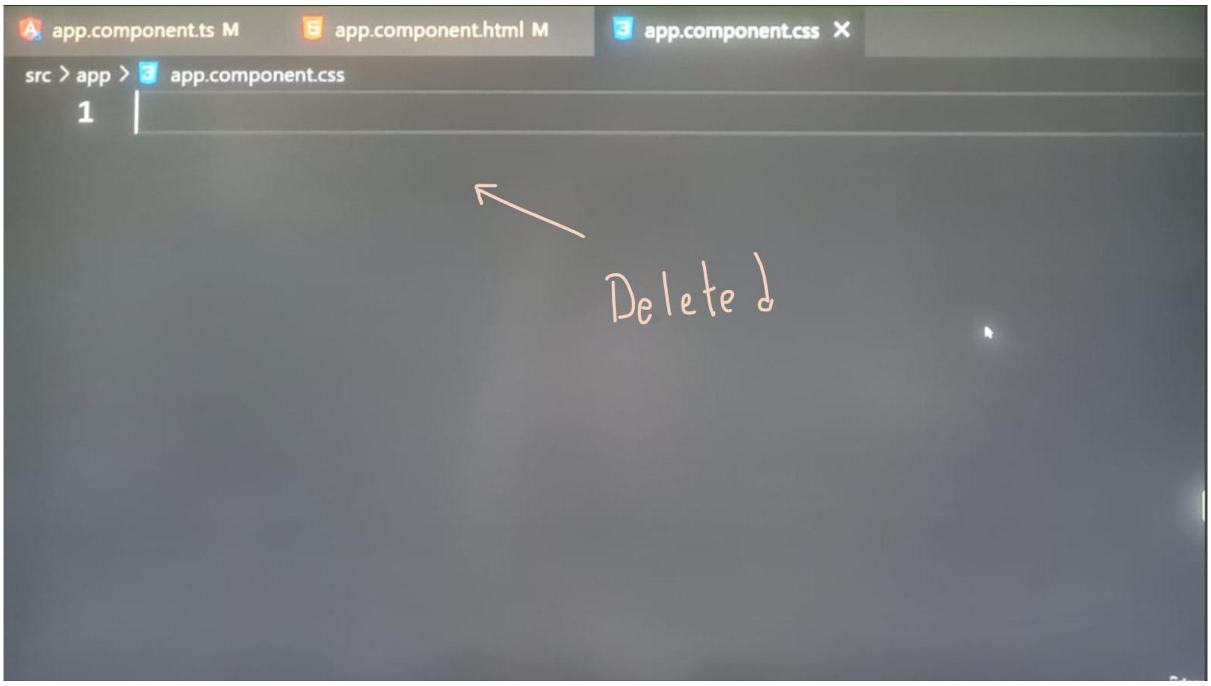
app.component.ts M · app.component.html M · app.component.css M ✕

src > app > app.component.css > p

```css
p {
  color: orange;
}
```

Delete

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

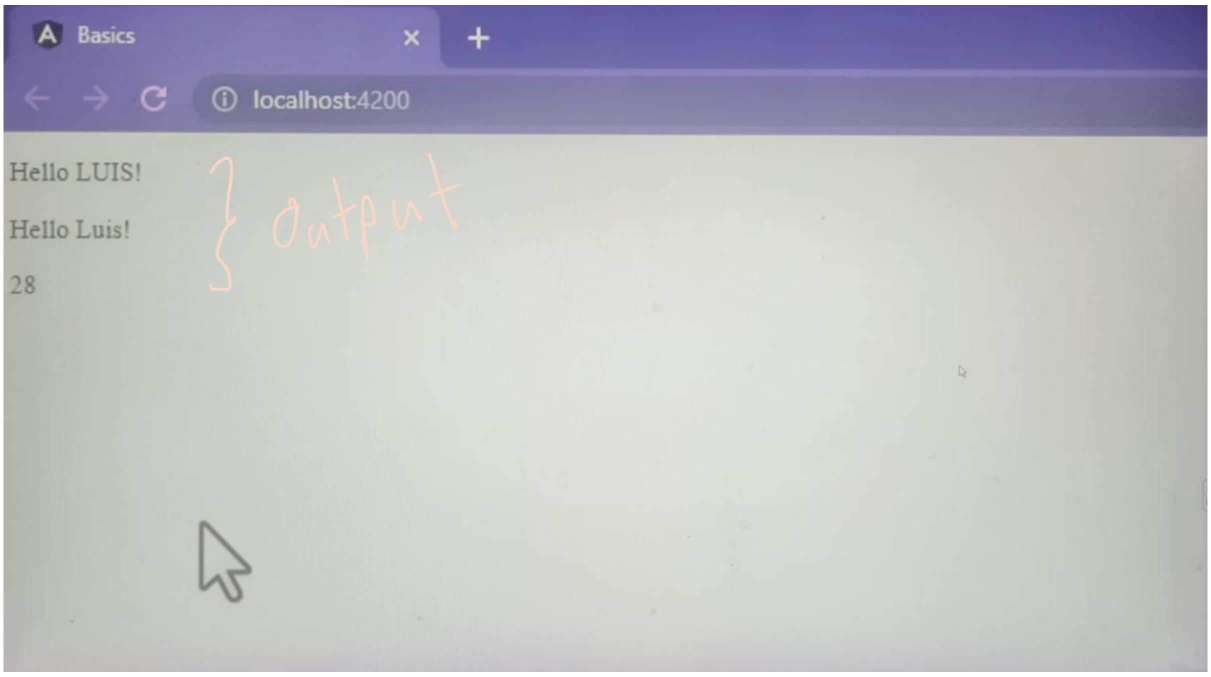


app.component.ts M · app.component.html M · app.component.css ✕

src > app > app.component.css

```css

```

Deleted



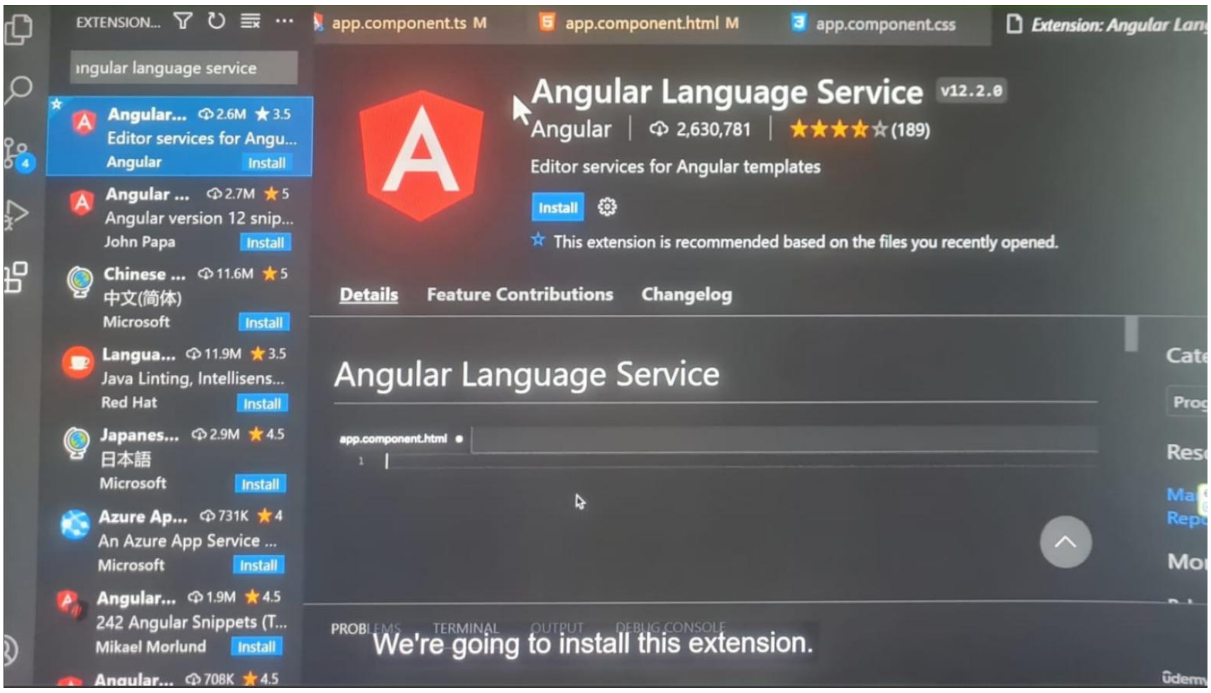
```
Hello LUIS!

Hello Luis!

28
```

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

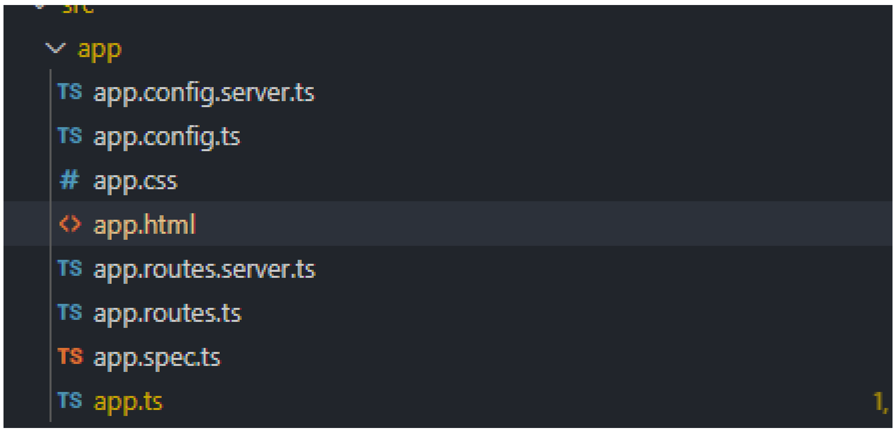
The code inside the curly braces must be an expression.

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

Angular will replace the curly braces with the evaluated value.

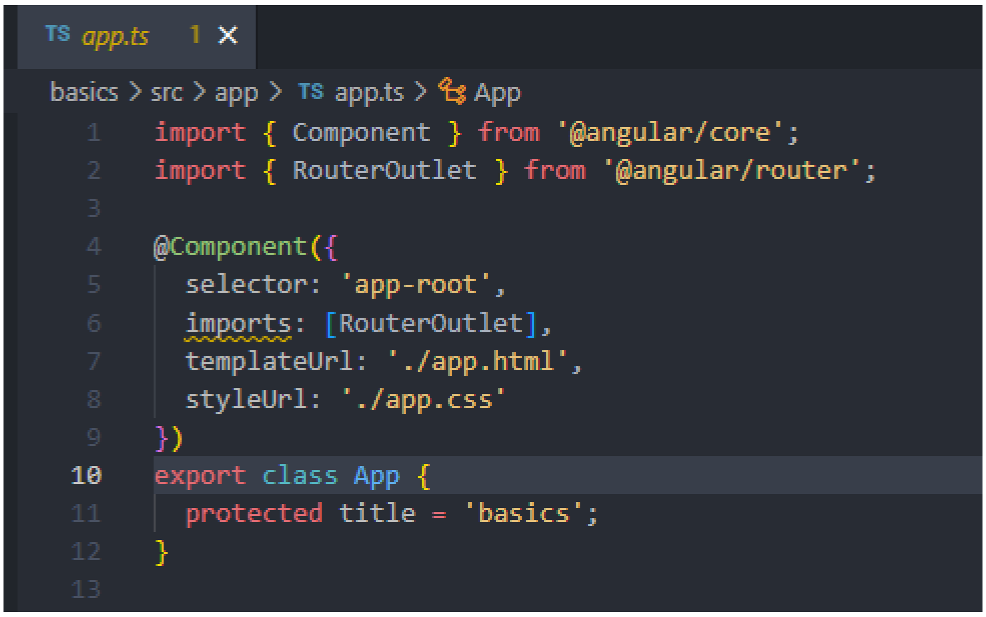

EXTENSION...
ingular language service
app.component.ts M
app.component.html M
app.component.css
Extension: Angular Lan
Angular Language Service v12.2.0
Angular | 2,630,781 | (189)
Editor services for Angular templates
Install
This extension is recommended based on the files you recently opened.
Details
Feature Contributions
Changelog
Angular Language Service
app.component.html
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE
We're going to install this extension.

On my PC with the latest Angular Version:



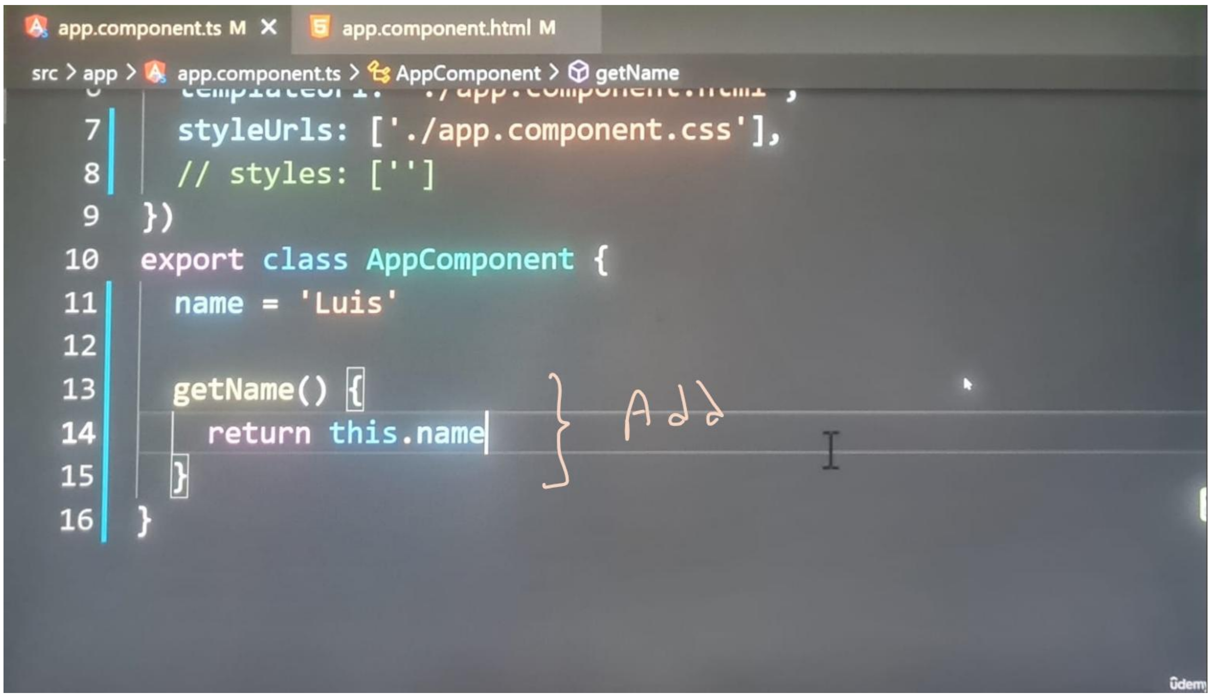
```
src
  app
    app.config.server.ts
    app.config.ts
    app.css
    app.html
    app.routes.server.ts
    app.routes.ts
    app.spec.ts
    app.ts
```

app.ts == app.component.ts:



```typescript
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App {
  protected title = 'basics';
}
```

This is what we did in the old version:

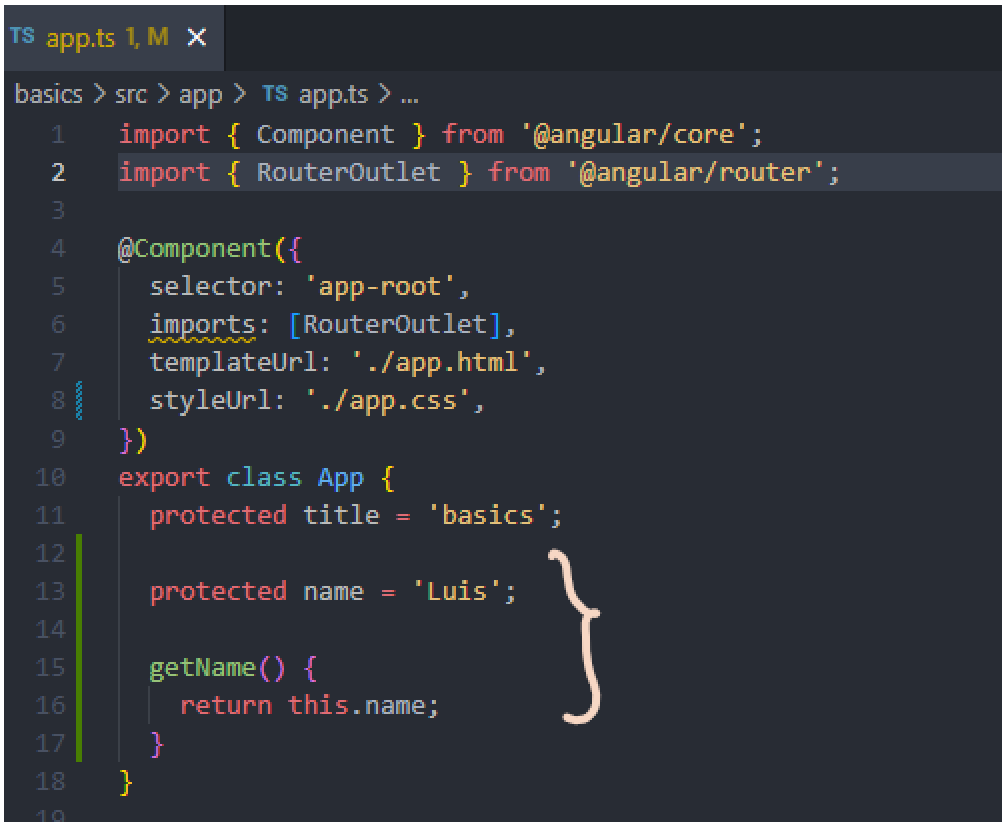


app.component.ts M × app.component.html M

src > app > app.component.ts > AppComponent > getName

```
  styleUrls: ['./app.component.css'],
  // styles: ['']
})
export class AppComponent {
  name = 'Luis'

  getName() {
    return this.name
  }
}
```

Now I will try to replicate that changes:



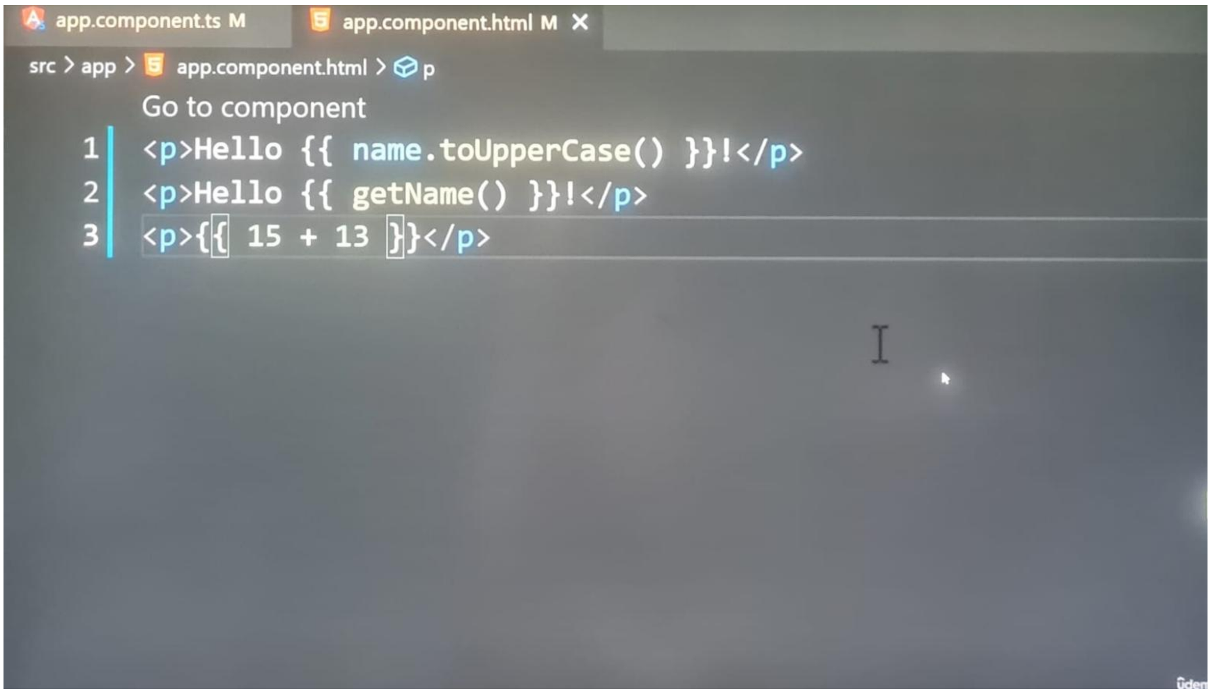
app.ts 1, M ×

basics > src > app > app.ts > ...

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrl: './app.css',
})
export class App {
  protected title = 'basics';

  protected name = 'Luis';

  getName() {
    return this.name;
  }
}
```

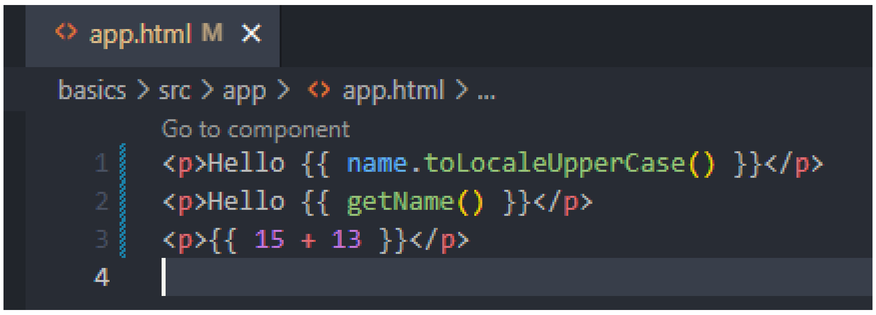
app.html == app.component.html:

Old version of app.component.html:



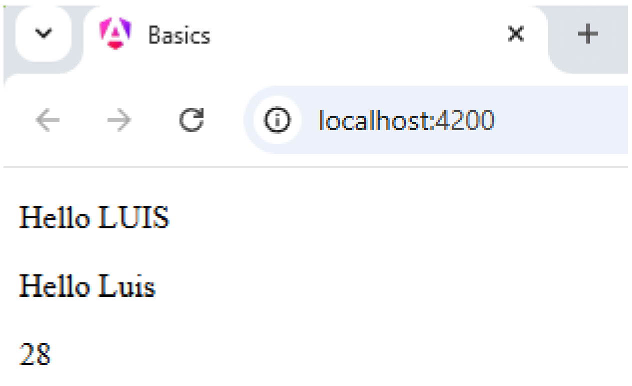
```
<p>Hello {{ name.toUpperCase() }}!</p>
<p>Hello {{ getName() }}!</p>
<p>{{ 15 + 13 }}</p>
```

My replication of the app.html:



```
<p>Hello {{ name.toLocaleUpperCase() }}</p>
<p>Hello {{ getName() }}</p>
<p>{{ 15 + 13 }}</p>

```

Output:



```
Hello LUIS

Hello Luis

28
```