Ng directives-Angular and Event Binding

22/4/24

NG Directives

"ng directives" likely refers to Angular directives. In Angular, directives are a way to extend the HTML vocabulary, creating custom elements and attributes that can add behavior and functionality to HTML elements. There are three types of directives in Angular:

1. Component Directives: These are the most common directives and are used to create reusable components. Components are directives with a template.
2. Attribute Directives: These are used to change the appearance or behavior of a DOM element, component, or another directive. They are applied as attributes to elements in the template.
3. Structural Directives: These alter the layout of the DOM by adding, removing, or manipulating elements. They are typically prefixed with an asterisk (*) and affect the structure of the DOM.

Component Directive

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

@Component({
  selector: 'app-custom-component',
  template: '<h1>Hello, {{ name }}</h1>'
})
export class CustomComponent {
  name = 'Angular';
}
```

In this example, `CustomComponent` is a directive with a template. It can be used as a custom HTML element `<app-custom-component></app-custom-component>`.

Structural Directive:

```
<div *ngIf="isLoggedIn; else notLoggedIn">

  Welcome, {{ username }}!

  <button (click)="logout()">Logout</button>

</div>

<ng-template #notLoggedIn>

  <button (click)="login()">Login</button>

</ng-template>
```

In this example, the `*ngIf` directive is a structural directive. It conditionally renders the content based on the value of `isLoggedIn`. If `isLoggedIn` is true, it displays the logged-in user's information; otherwise, it shows a login button. The `else` keyword with `ng-template` provides an alternative template for the `*ngIf` directiv

NGSTYLE -WORKING

Step 1- First of all, you have to create one component

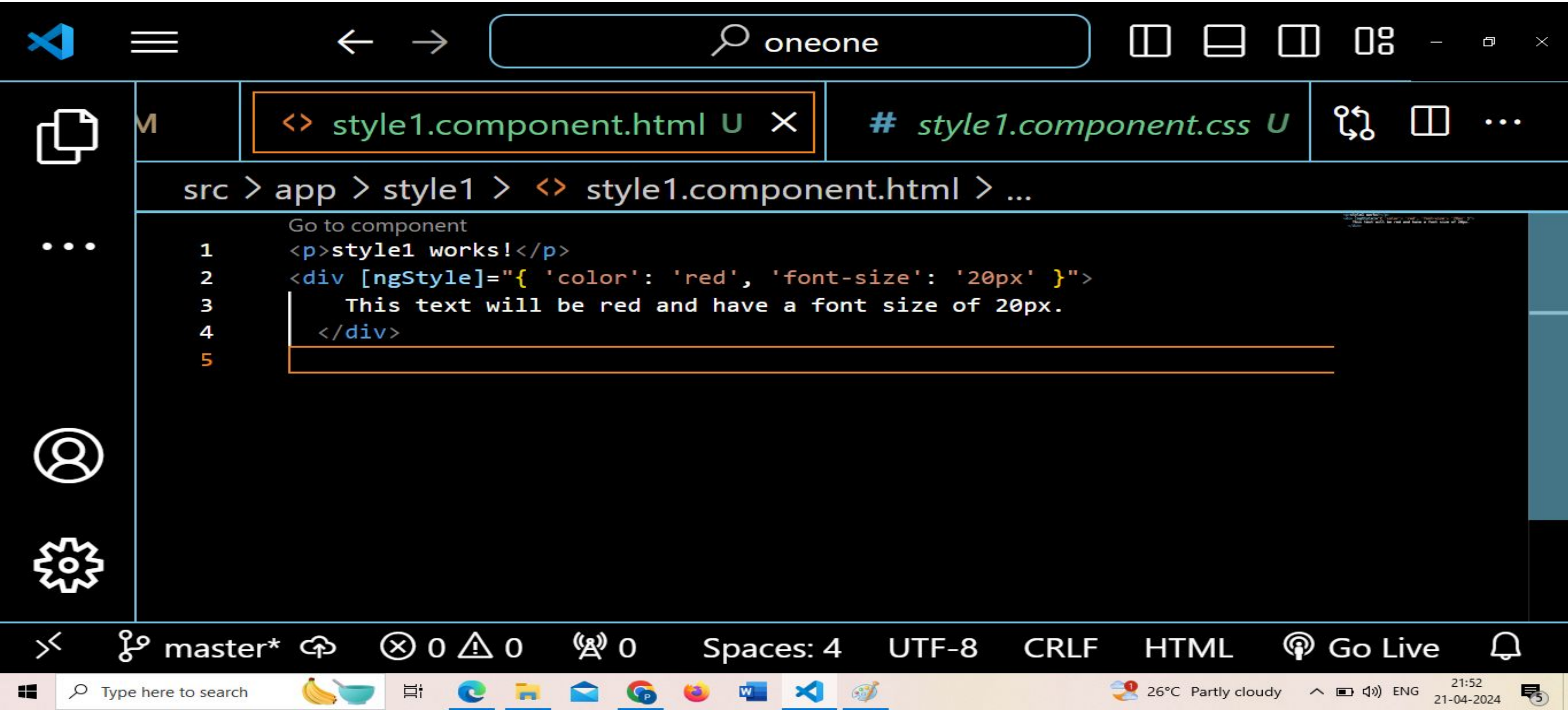
Ng g c style1

Then you will be able to see four file created style1.component.html,style1.component.ts

Style1.component.css

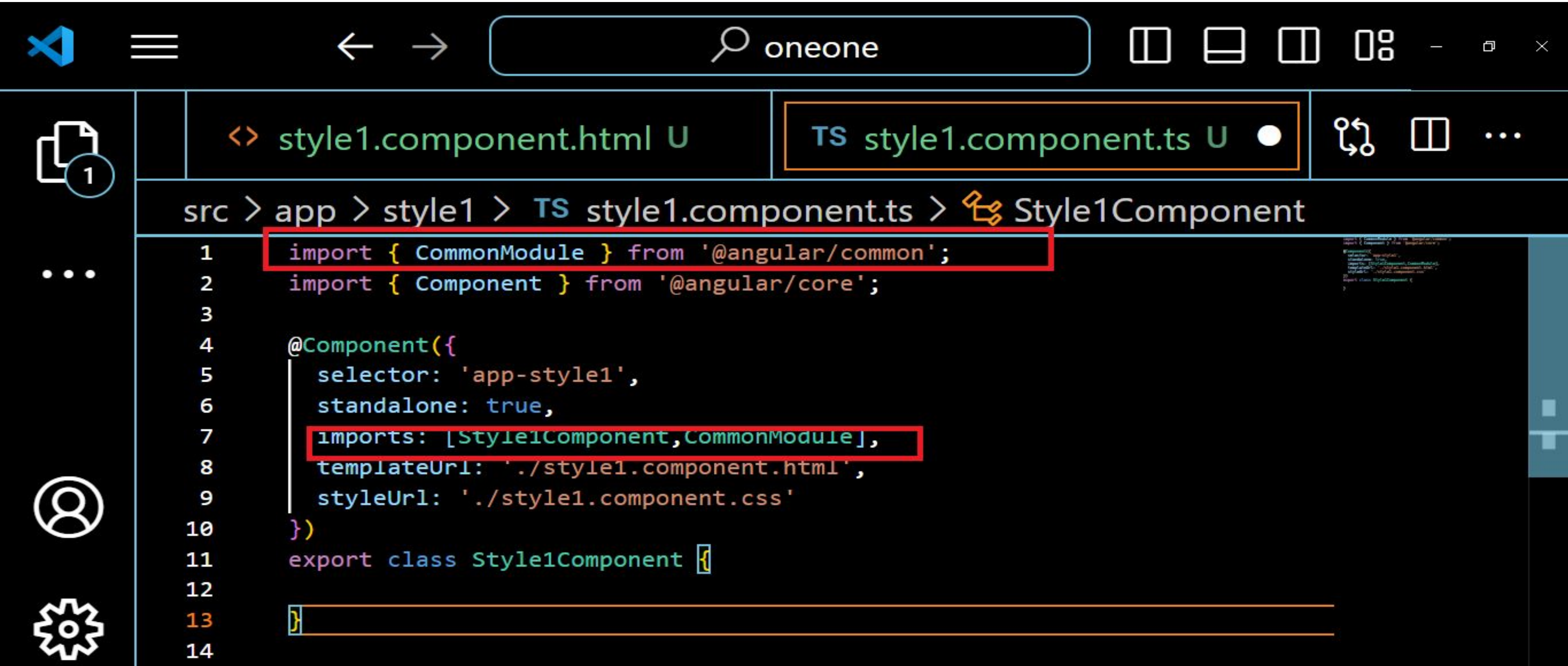
style1.component.spec.ts

Make use of [ngStyle] in your component.html(created component)



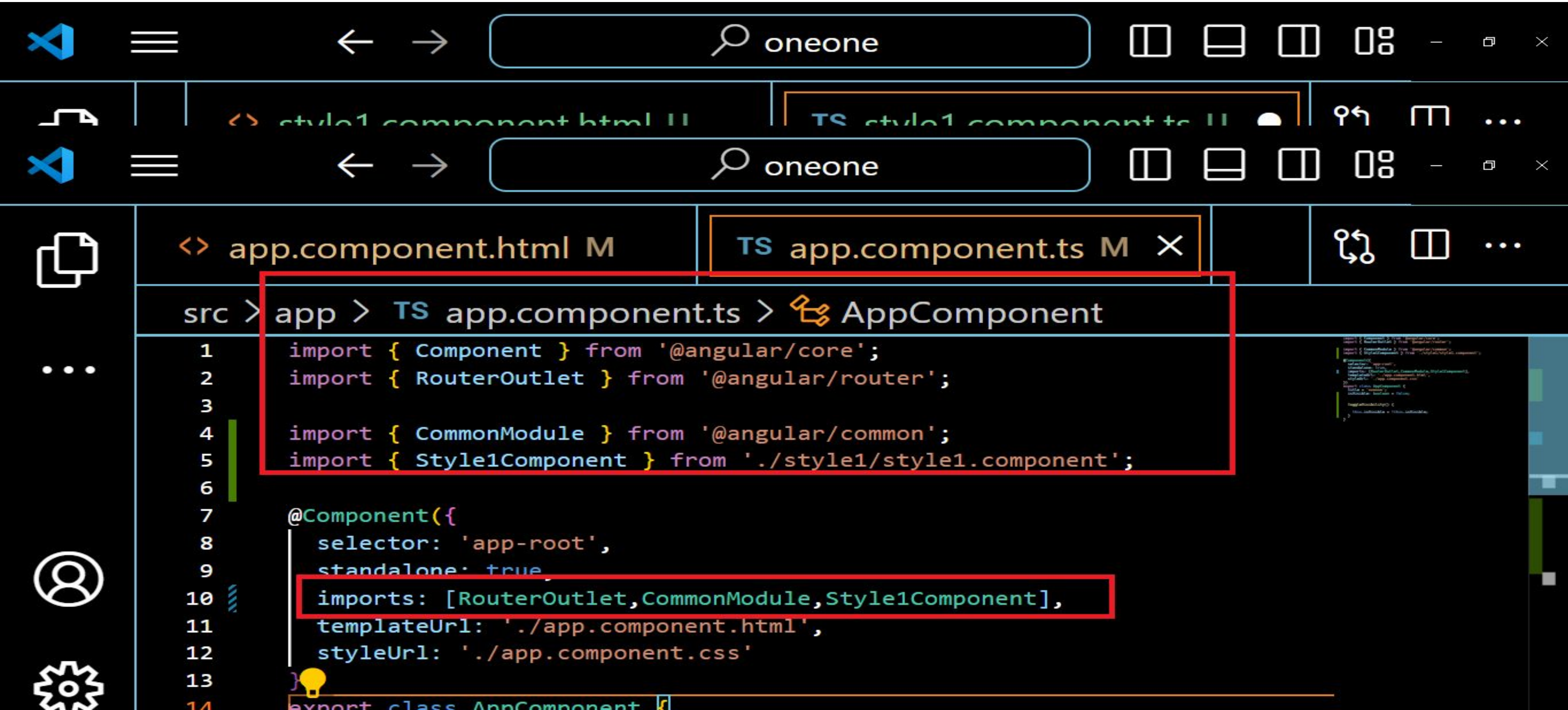
```html
<p>style1 works!</p>
<div [ngStyle]="{ 'color': 'red', 'font-size': '20px' }">
    This text will be red and have a font size of 20px.
  </div>
```

Import common module and compone


oneone
style1.component.html U
TS style1.component.ts U
src > app > style1 > TS style1.component.ts > Style1Component

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

@Component({
  selector: 'app-style1',
  standalone: true,
  imports: [Style1Component,CommonModule],
  templateUrl: './style1.component.html',
  styleUrl: './style1.component.css'
})
export class Style1Component {

}
```

Open app.component.ts

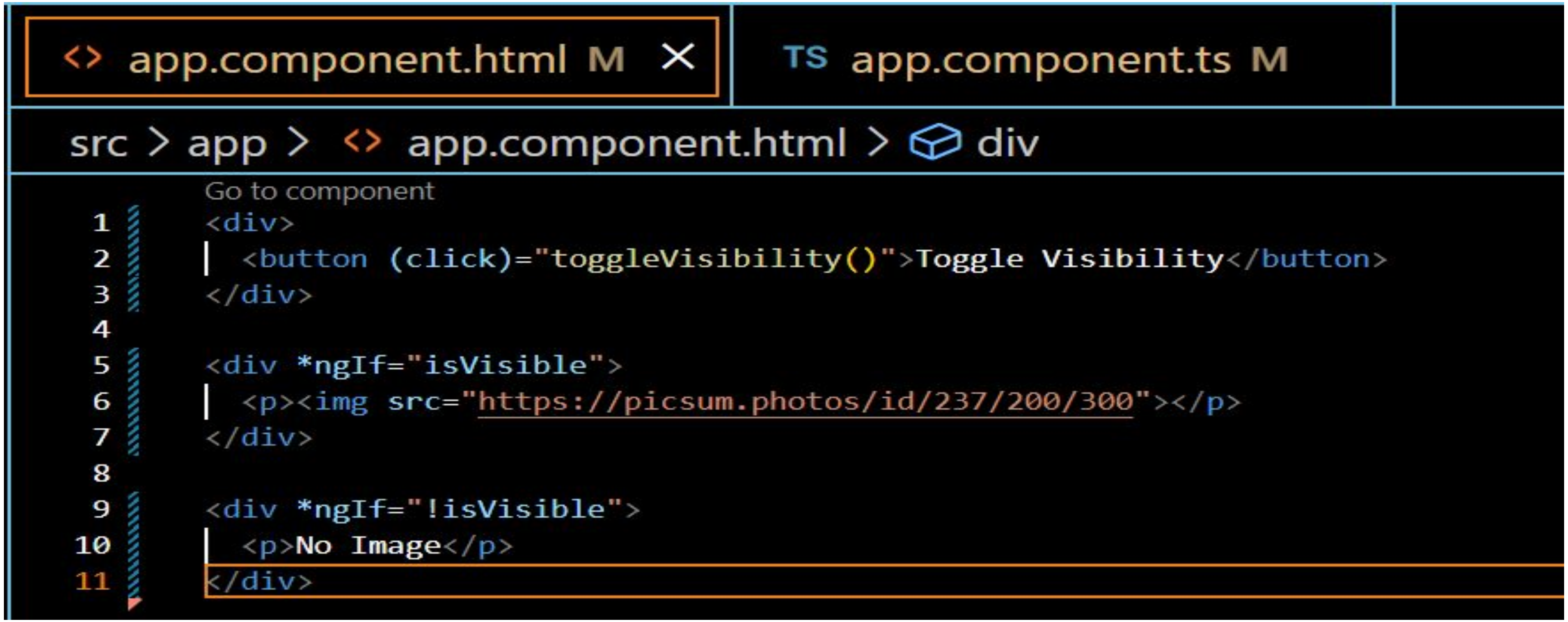


```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

import { CommonModule } from '@angular/common';
import { Style1Component } from './style1/style1.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet,CommonModule,Style1Component],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
```

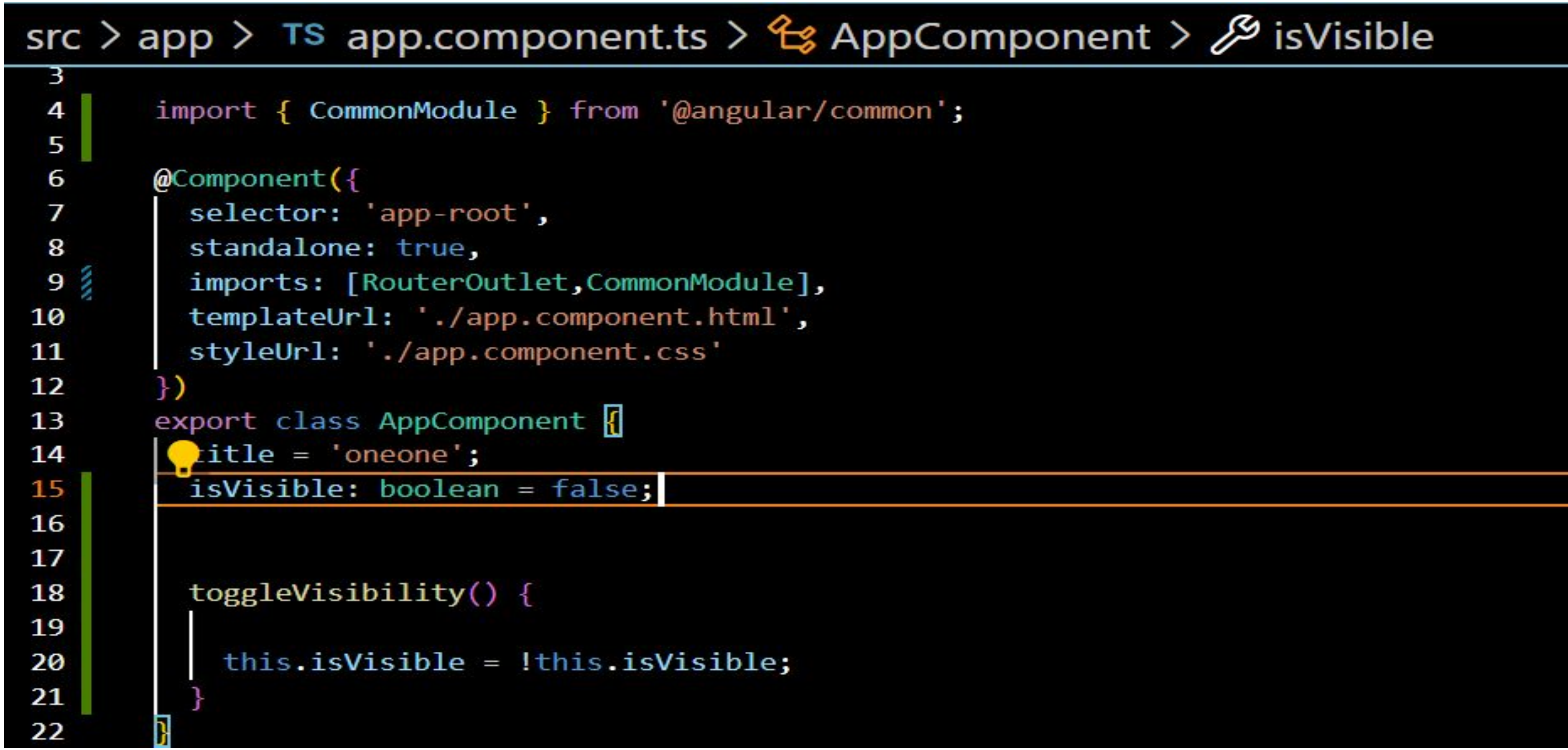
style1 works!

This text will be red and have a font size of 20px.

Directive-NgIf

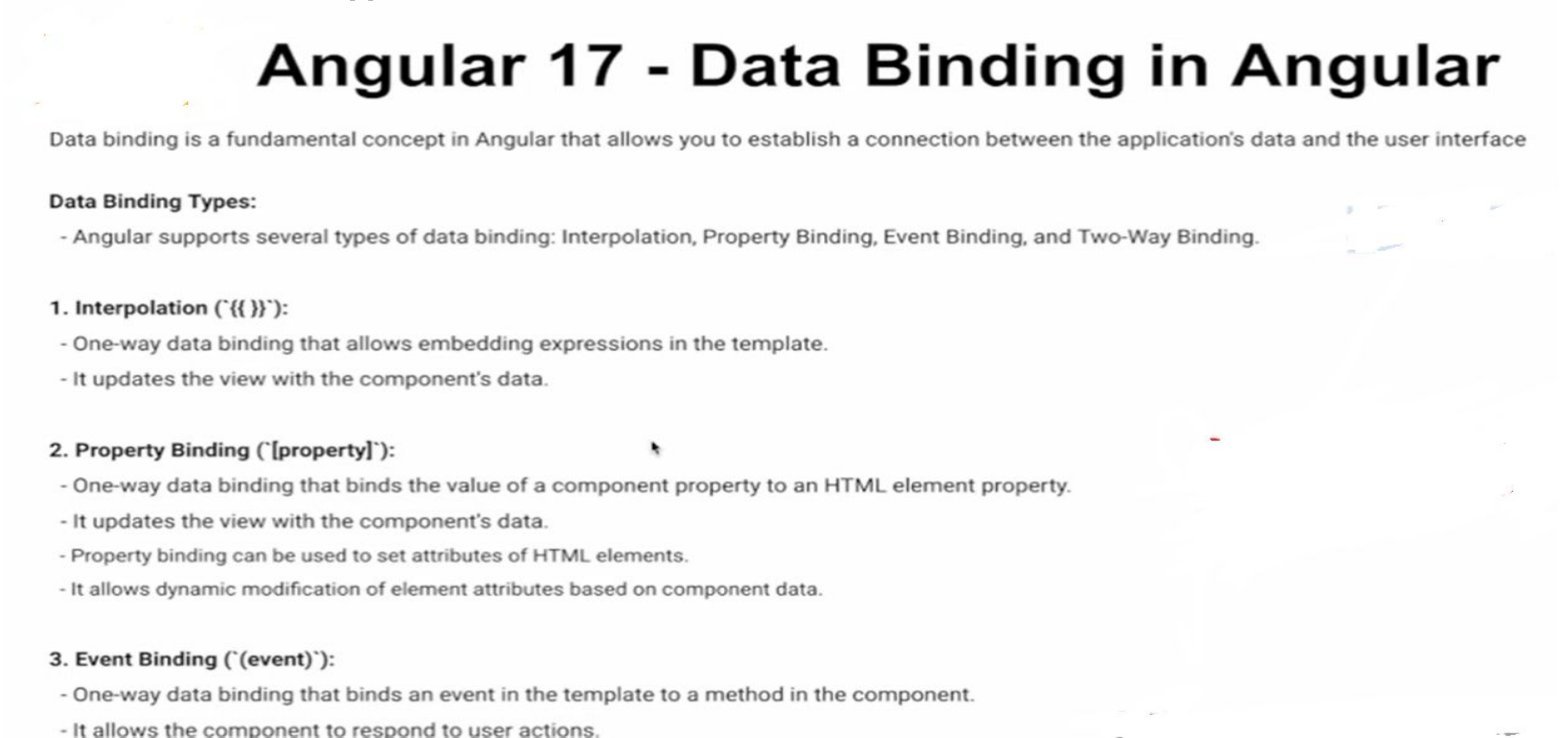


```html
<div>
  <button (click)="toggleVisibility()">Toggle Visibility</button>
</div>

<div *ngIf="isVisible">
  <p><img src="https://picsum.photos/id/237/200/300"></p>
</div>

<div *ngIf="!isVisible">
  <p>No Image</p>
</div>
```


src > app > TS app.component.ts > AppComponent > isVisible

```
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet,CommonModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'oneone';
  isVisible: boolean = false;


  toggleVisibility() {

    this.isVisible = !this.isVisible;
  }
}
```

Data Binding



Angular 17 - Data Binding in Angular

Data binding is a fundamental concept in Angular that allows you to establish a connection between the application's data and the user interface

Data Binding Types:

- Angular supports several types of data binding: Interpolation, Property Binding, Event Binding, and Two-Way Binding.

1. Interpolation (`{{ }}`):

- One-way data binding that allows embedding expressions in the template.
- It updates the view with the component's data.

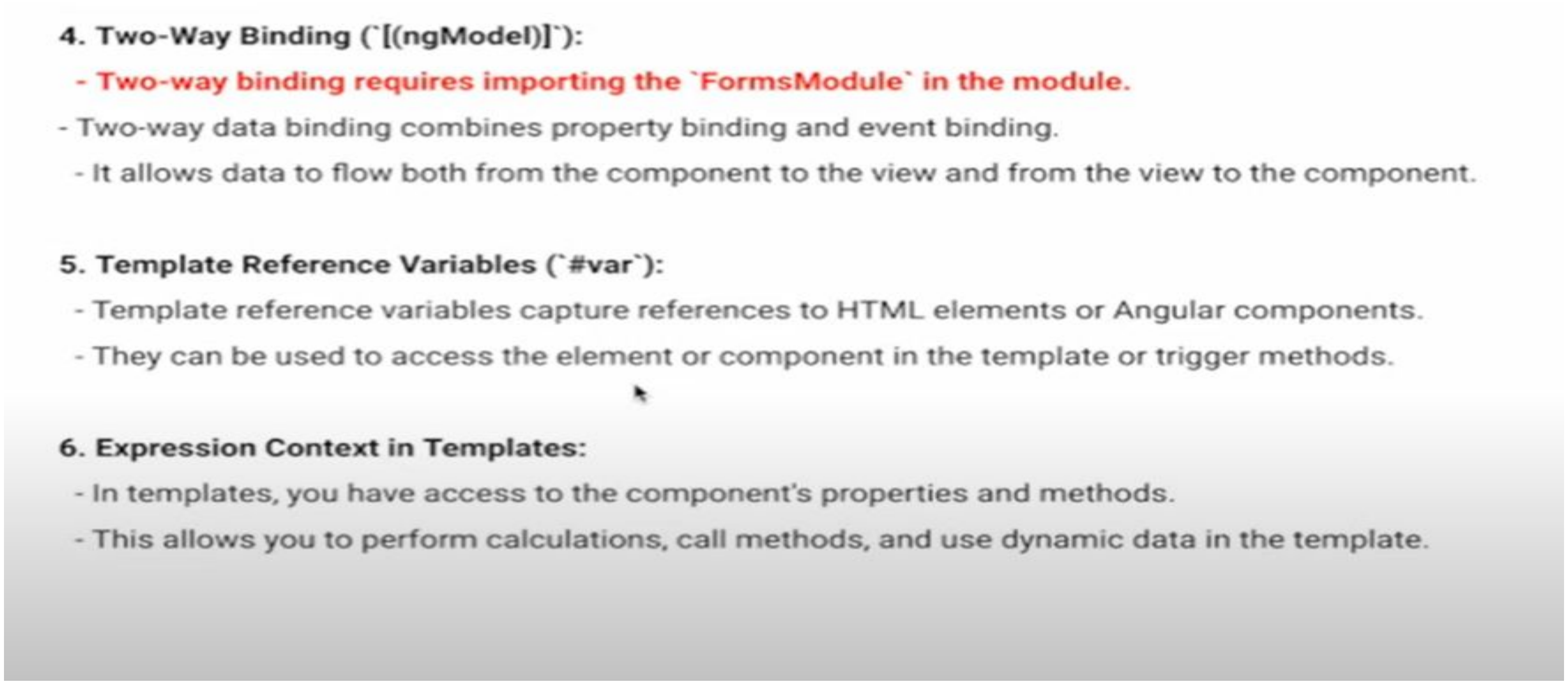
2. Property Binding (`[property]`):

- One-way data binding that binds the value of a component property to an HTML element property.
- It updates the view with the component's data.
- Property binding can be used to set attributes of HTML elements.
- It allows dynamic modification of element attributes based on component data.

3. Event Binding (`(event)`):

- One-way data binding that binds an event in the template to a method in the component.
- It allows the component to respond to user actions.

Two way binding



4. Two-Way Binding (`[(ngModel)]`):

- Two-way binding requires importing the `FormsModule` in the module.

- Two-way data binding combines property binding and event binding.
- It allows data to flow both from the component to the view and from the view to the component.

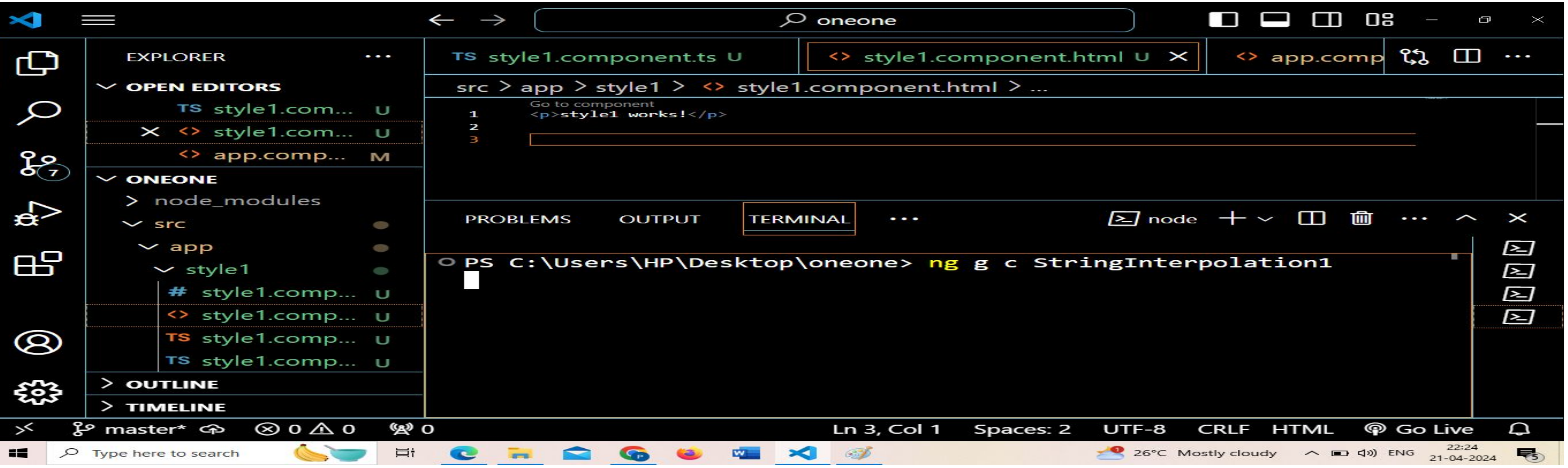
5. Template Reference Variables (`#var`):

- Template reference variables capture references to HTML elements or Angular components.
- They can be used to access the element or component in the template or trigger methods.

6. Expression Context in Templates:

- In templates, you have access to the component's properties and methods.
- This allows you to perform calculations, call methods, and use dynamic data in the template.

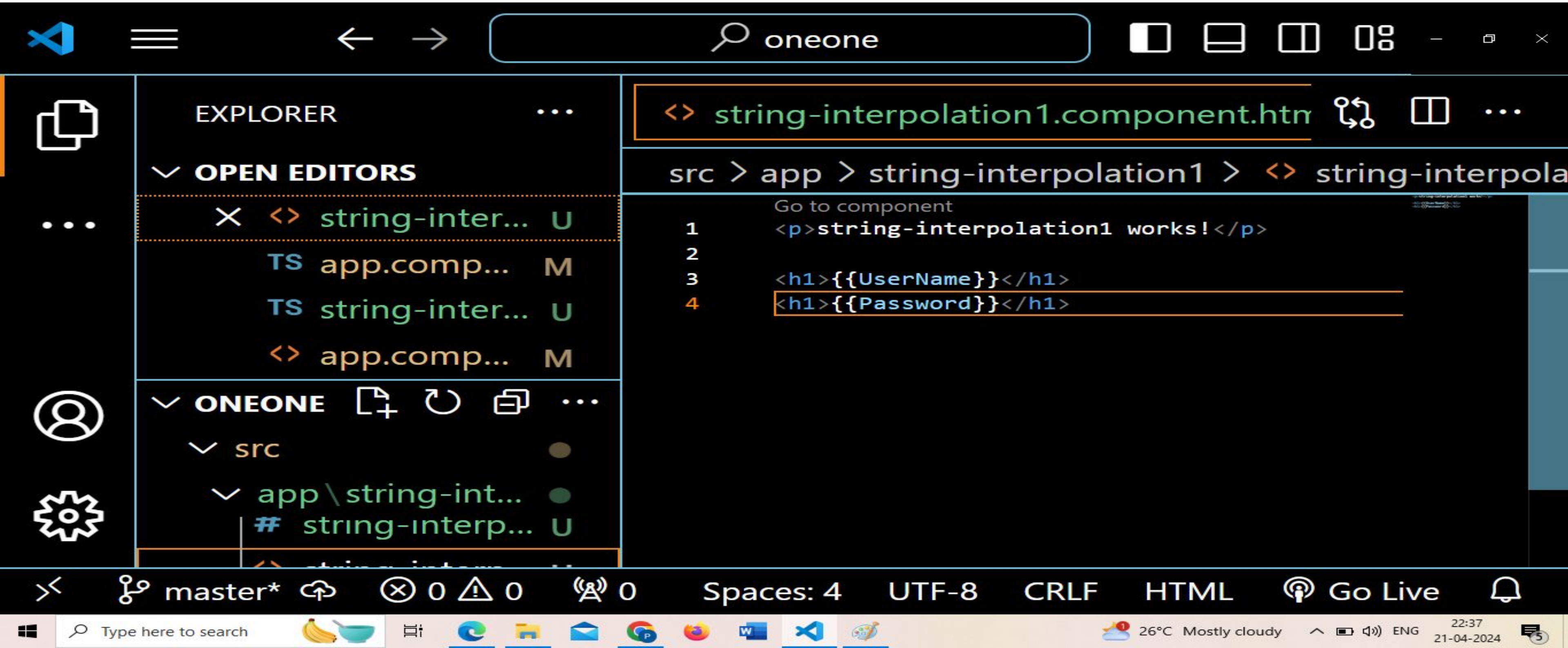
Create one component first, its your choice whether you want to create component with –no-standalone or either you can create directly as well



```
Go to component
<p>style1 works!</p>
```

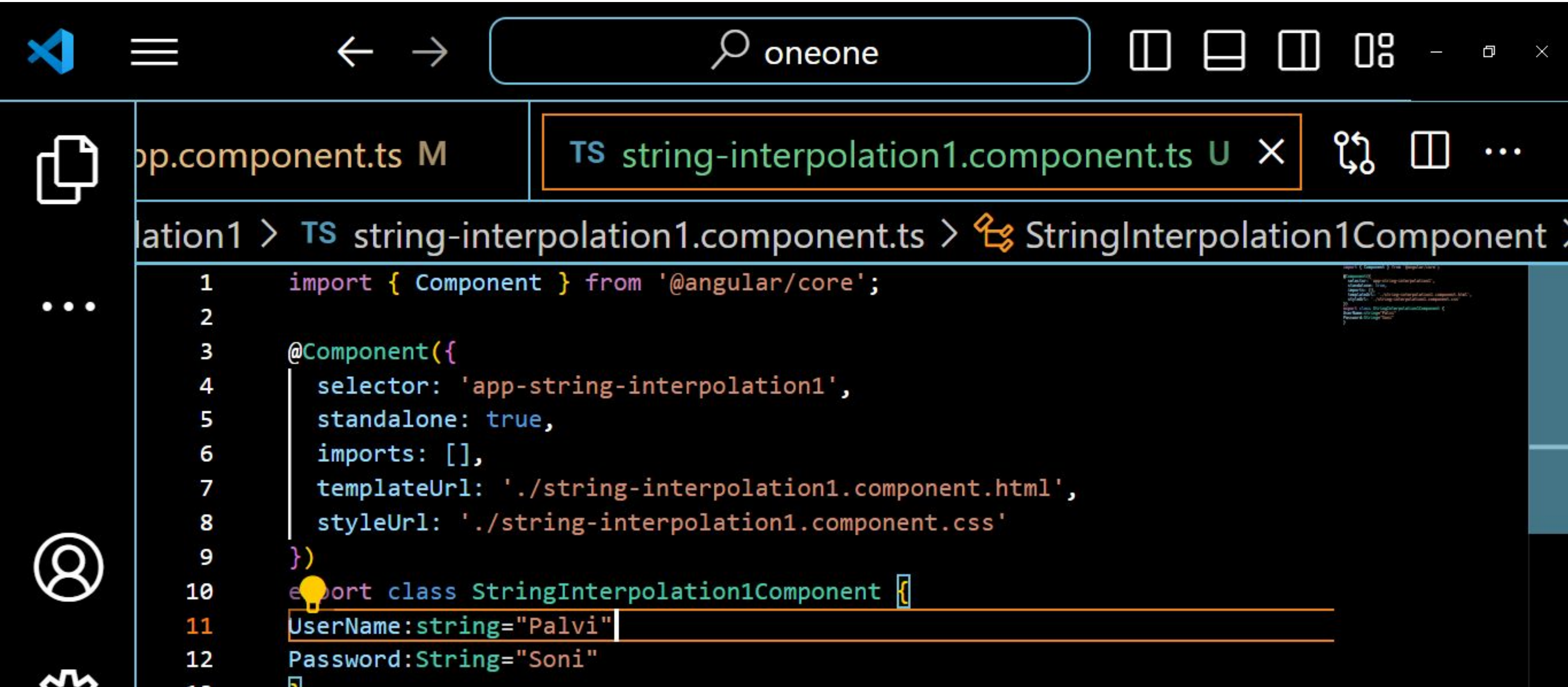
```
PS C:\Users\HP\Desktop\oneone> ng g c StringInterpolation1
```

Open stringinterpolation.component.html



```html
<p>string-interpolation1 works!</p>

<h1>{{UserName}}</h1>
<h1>{{Password}}</h1>
```

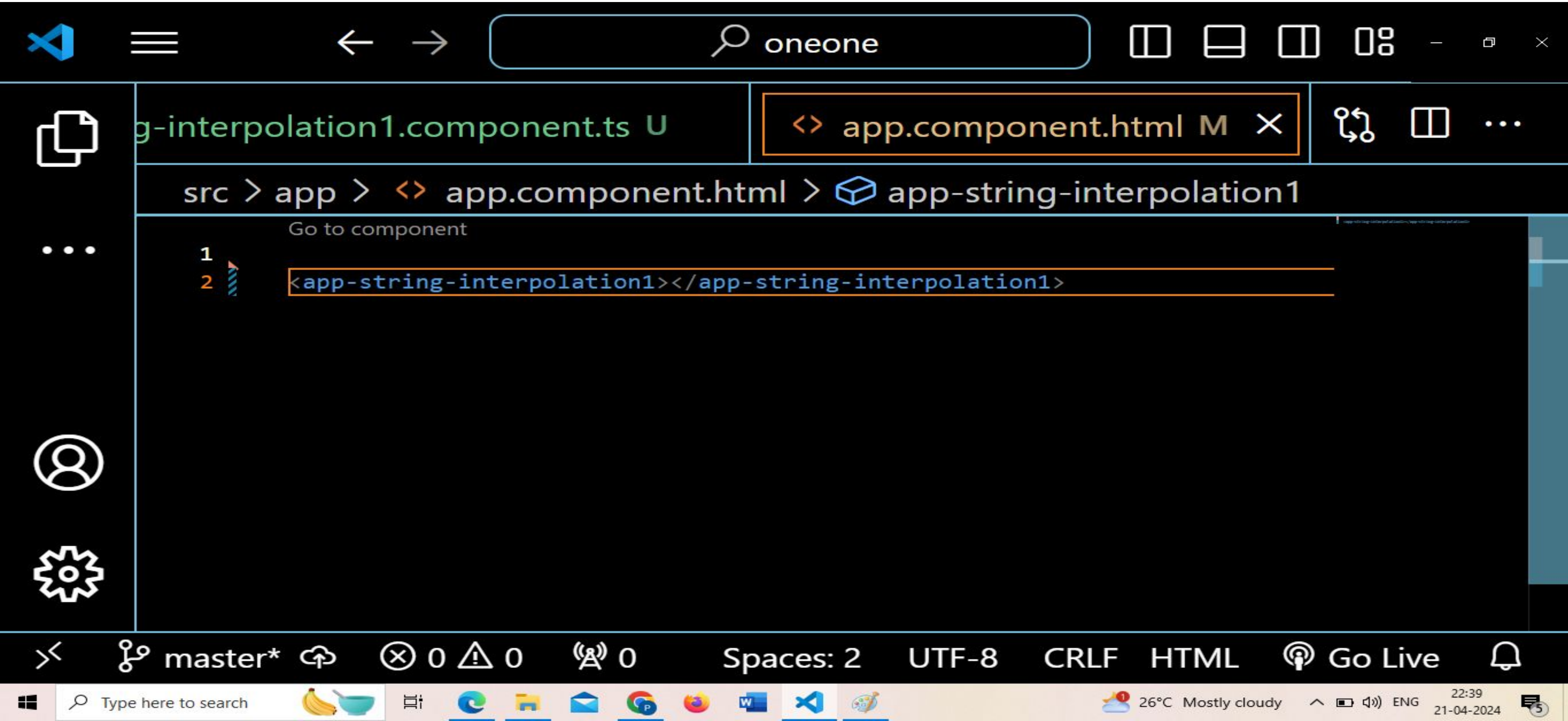
Open stringinterpolation.component.ts



oneone
pp.component.ts M
TS string-interpolation1.component.ts U
lation1 > TS string-interpolation1.component.ts > StringInterpolation1Component

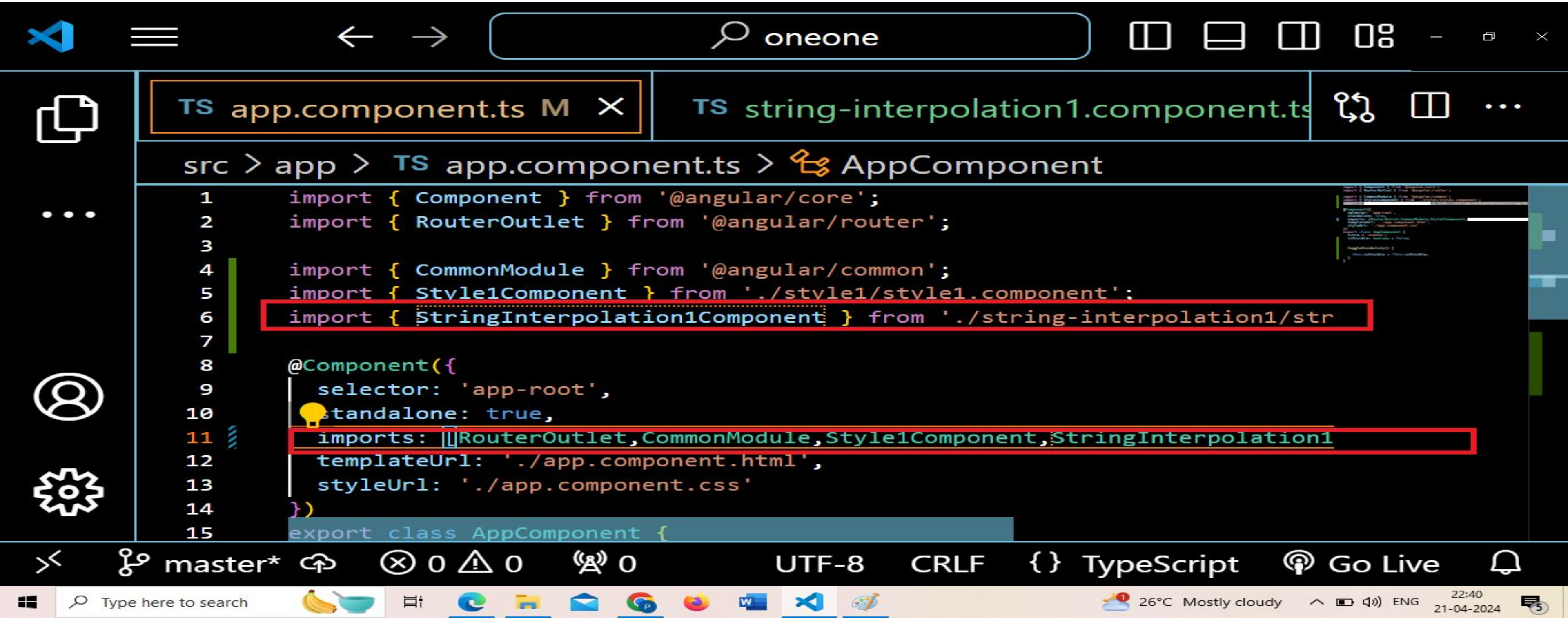
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-string-interpolation1',
  standalone: true,
  imports: [],
  templateUrl: './string-interpolation1.component.html',
  styleUrl: './string-interpolation1.component.css'
})
export class StringInterpolation1Component {
UserName:string="Palvi"
Password:String="Soni"
```

Open app.component.html



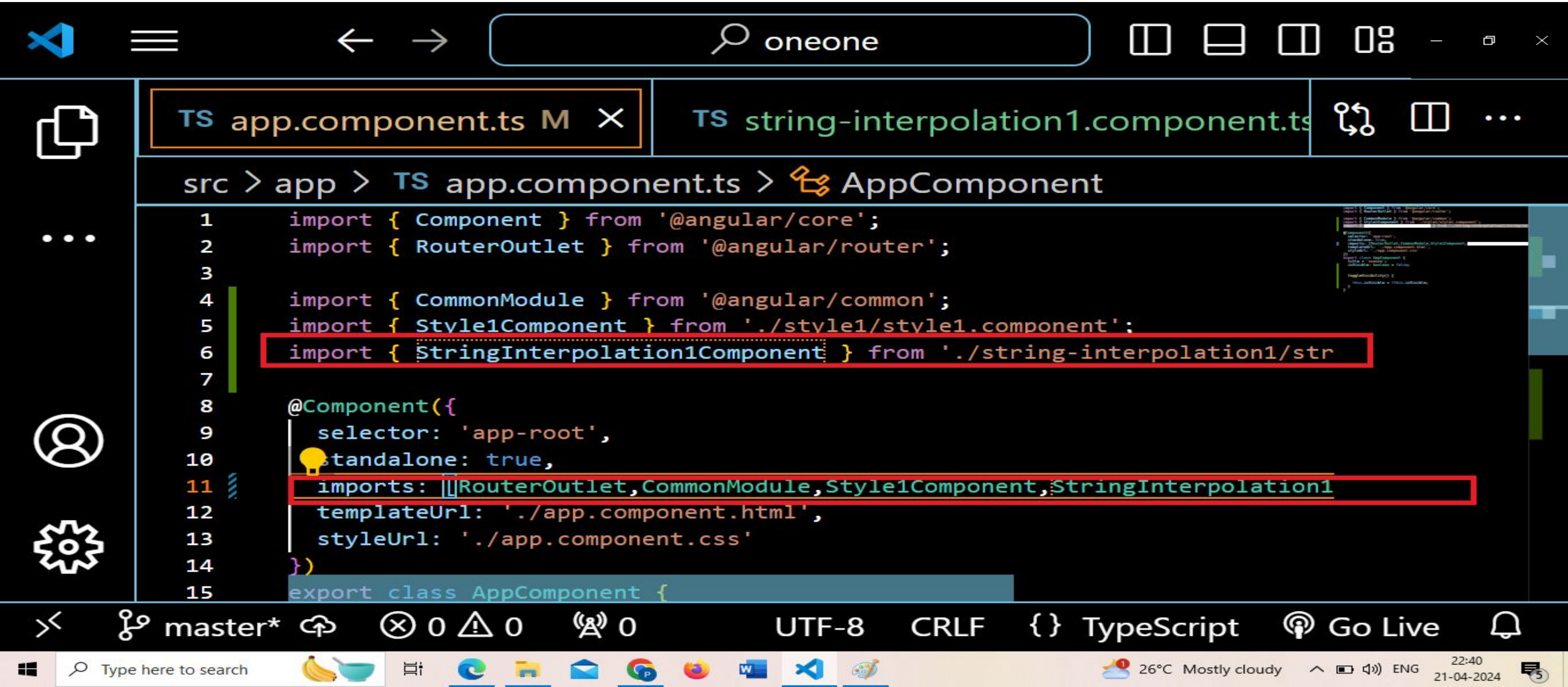
```html

<app-string-interpolation1></app-string-interpolation1>
```

Open app.component.ts- import the necessary components



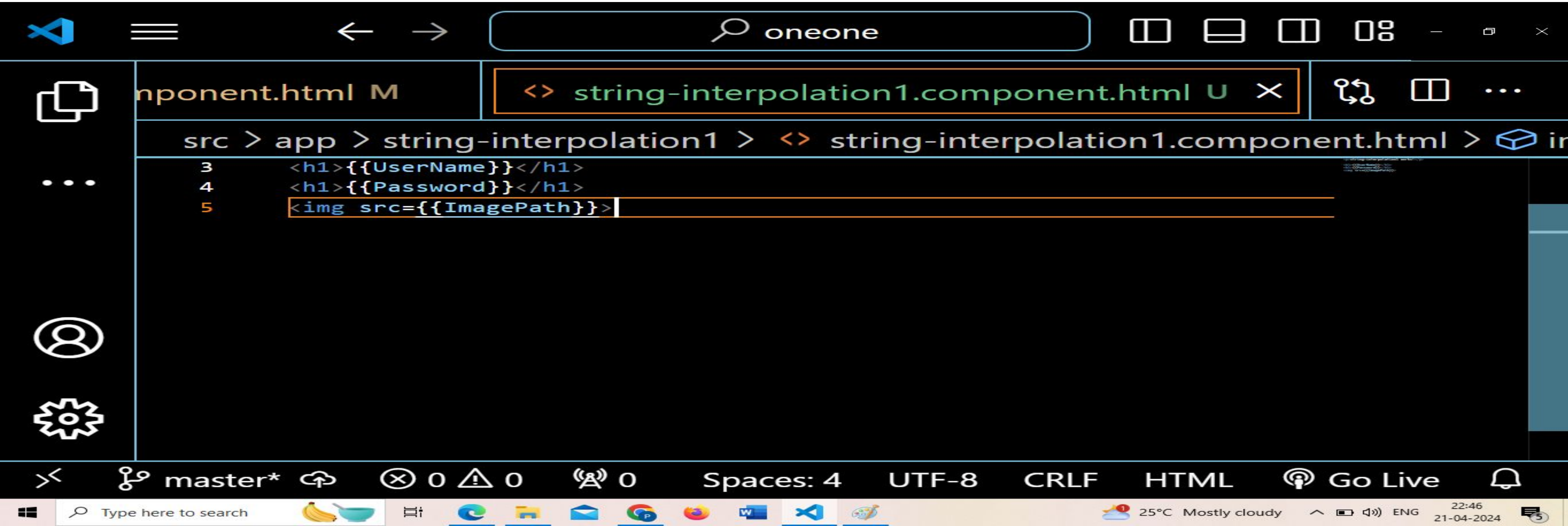
```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

import { CommonModule } from '@angular/common';
import { Style1Component } from './style1/style1.component';
import { StringInterpolation1Component } from './string-interpolation1/str

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet,CommonModule,Style1Component,StringInterpolation1
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
```

output

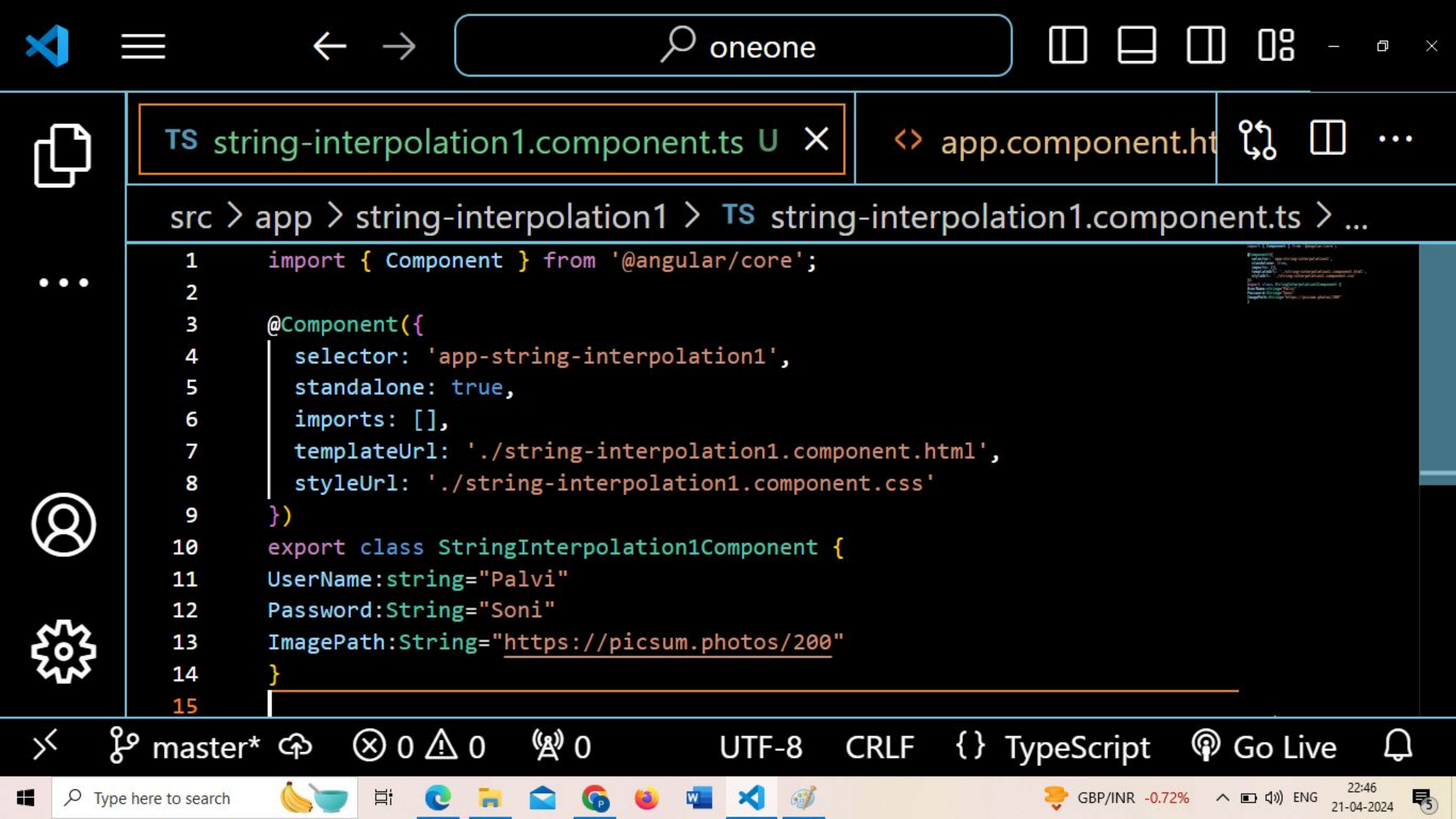


```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

import { CommonModule } from '@angular/common';
import { Style1Component } from './style1/style1.component';
import { StringInterpolation1Component } from './string-interpolation1/str

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet,CommonModule,Style1Component,StringInterpolation1
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
```

String Interpolation for image



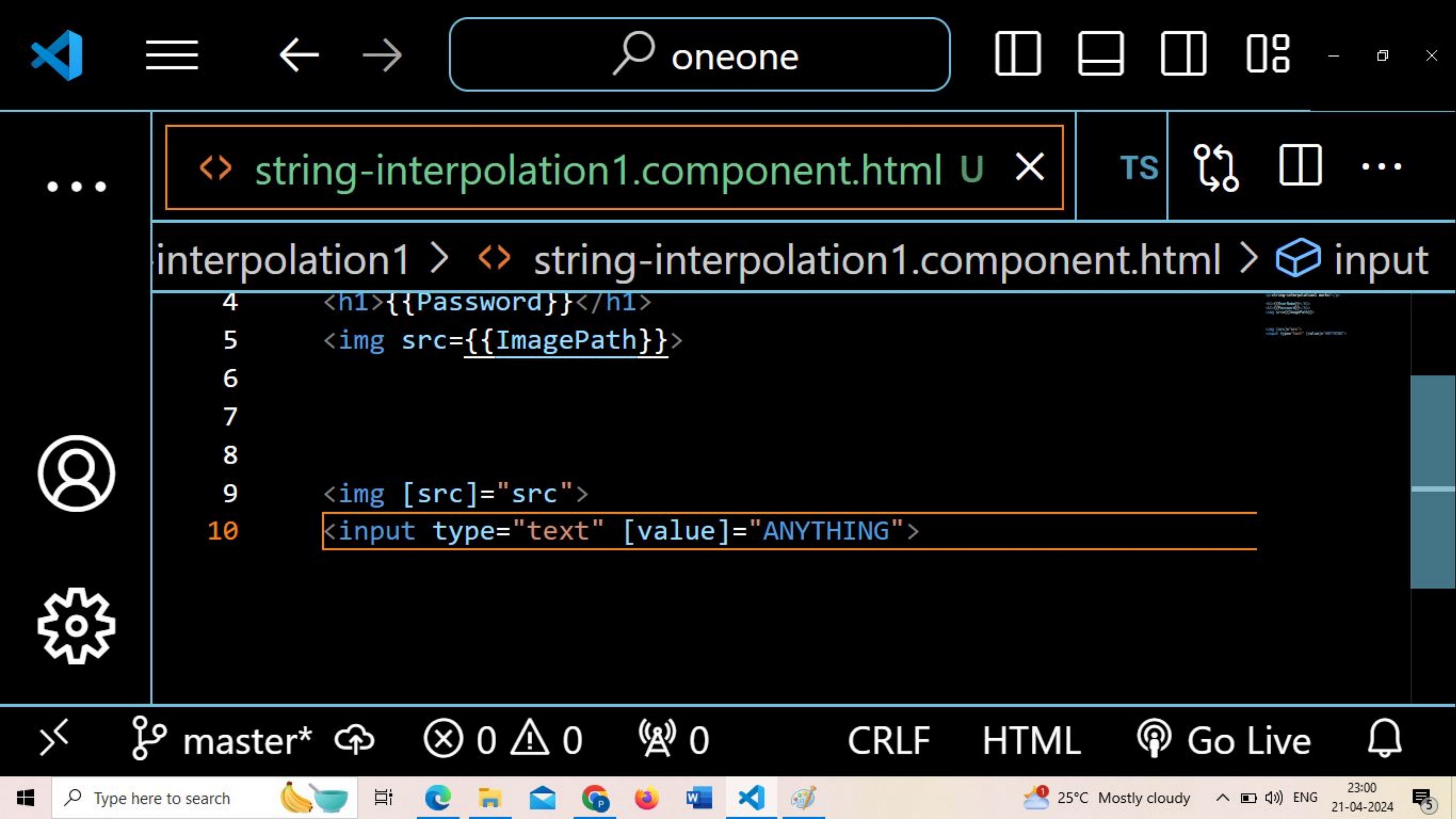
```
<h1>{{UserName}}</h1>
<h1>{{Password}}</h1>
<img src={{ImagePath}}>
```


oneone
TS string-interpolation1.component.ts U
app.component.ht
src > app > string-interpolation1 > TS string-interpolation1.component.ts > ...
master*
0 0
0
UTF-8
CRLF
TypeScript
Go Live
Type here to search
GBP/INR -0.72%
ENG
22:46
21-04-2024

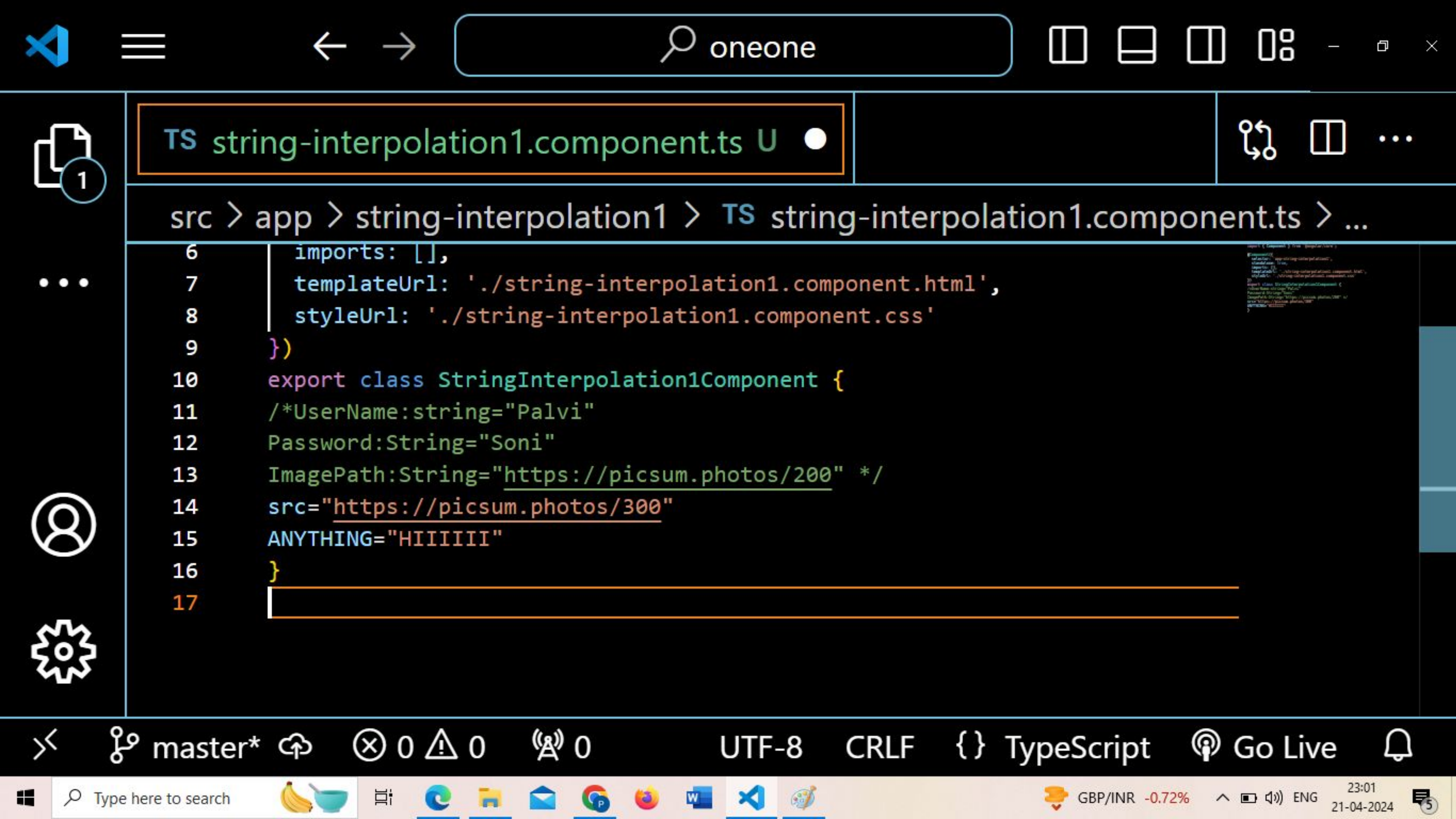
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-string-interpolation1',
  standalone: true,
  imports: [],
  templateUrl: './string-interpolation1.component.html',
  styleUrl: './string-interpolation1.component.css'
})
export class StringInterpolation1Component {
UserName:string="Palvi"
Password:String="Soni"
ImagePath:String="https://picsum.photos/200"
}

```

Property Binding

Property Binding is a **one-way data-binding** technique. In property binding, we bind a property of a DOM element to a field which is a defined property in our component TypeScript code. Actually, Angular internally converts string interpolation into property binding.

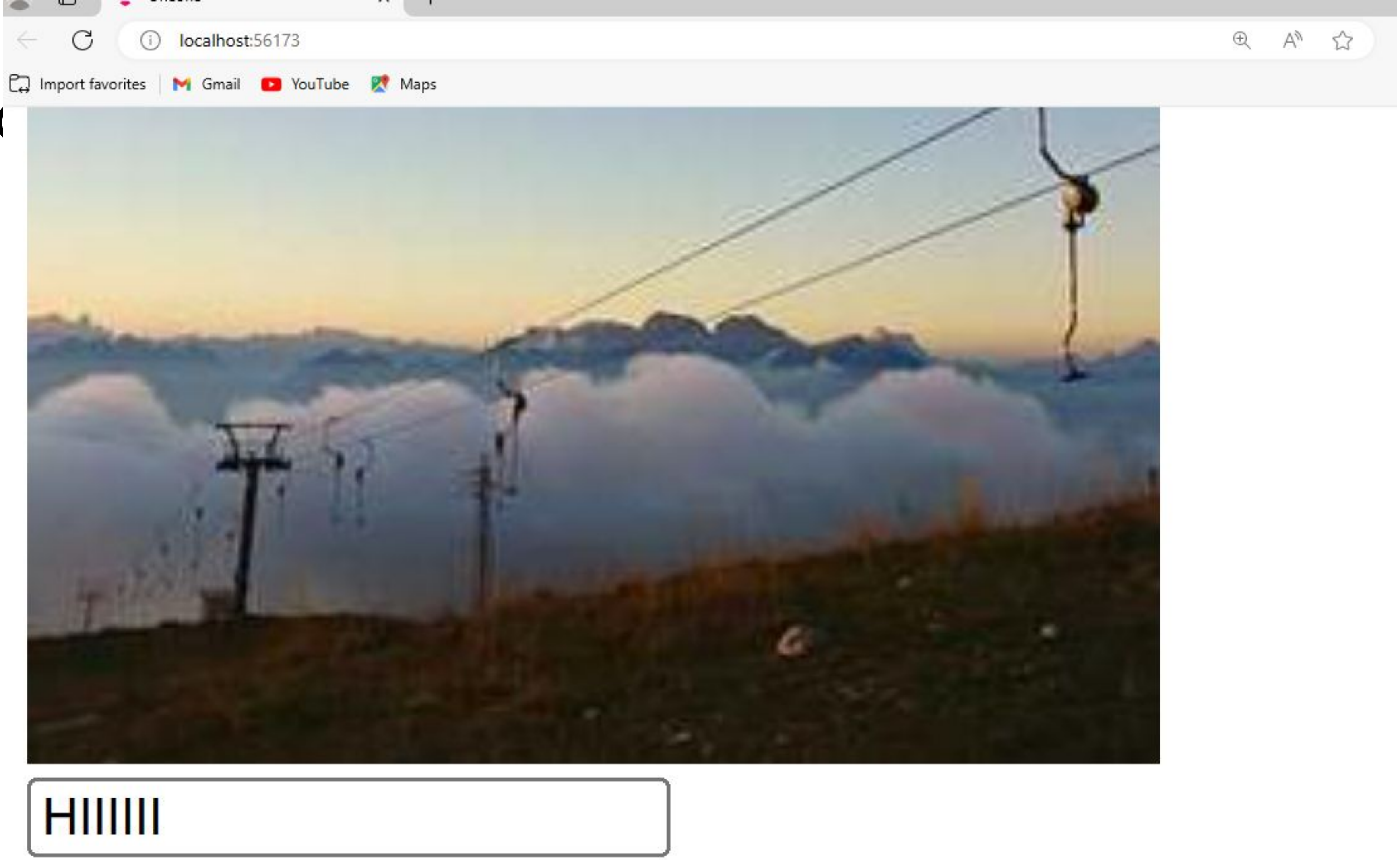

oneone
string-interpolation1.component.html U
TS
interpolation1 > string-interpolation1.component.html > input
master*
0 0
0
CRLF
HTML
Go Live
Type here to search
25°C Mostly cloudy
ENG
23:00
21-04-2024

```html
<h1>{{Password}}</h1>
<img src={{ImagePath}}>



<img [src]="src">
<input type="text" [value]="ANYTHING">
```

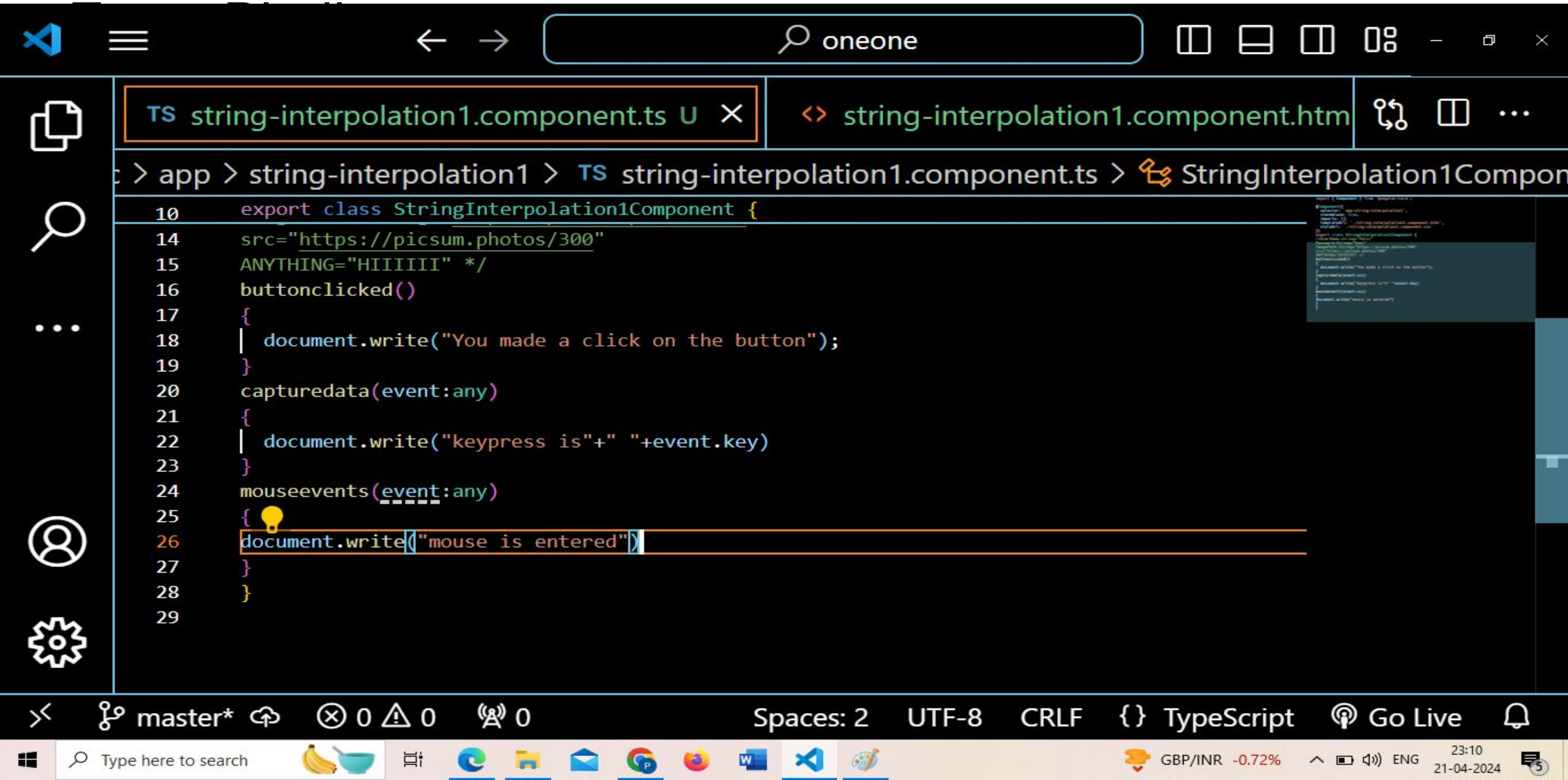


oneone
TS string-interpolation1.component.ts U
src > app > string-interpolation1 > TS string-interpolation1.component.ts > ...
master*
0 0
0
UTF-8
CRLF
{} TypeScript
Go Live
Type here to search
GBP/INR -0.72%
ENG
23:01
21-04-2024

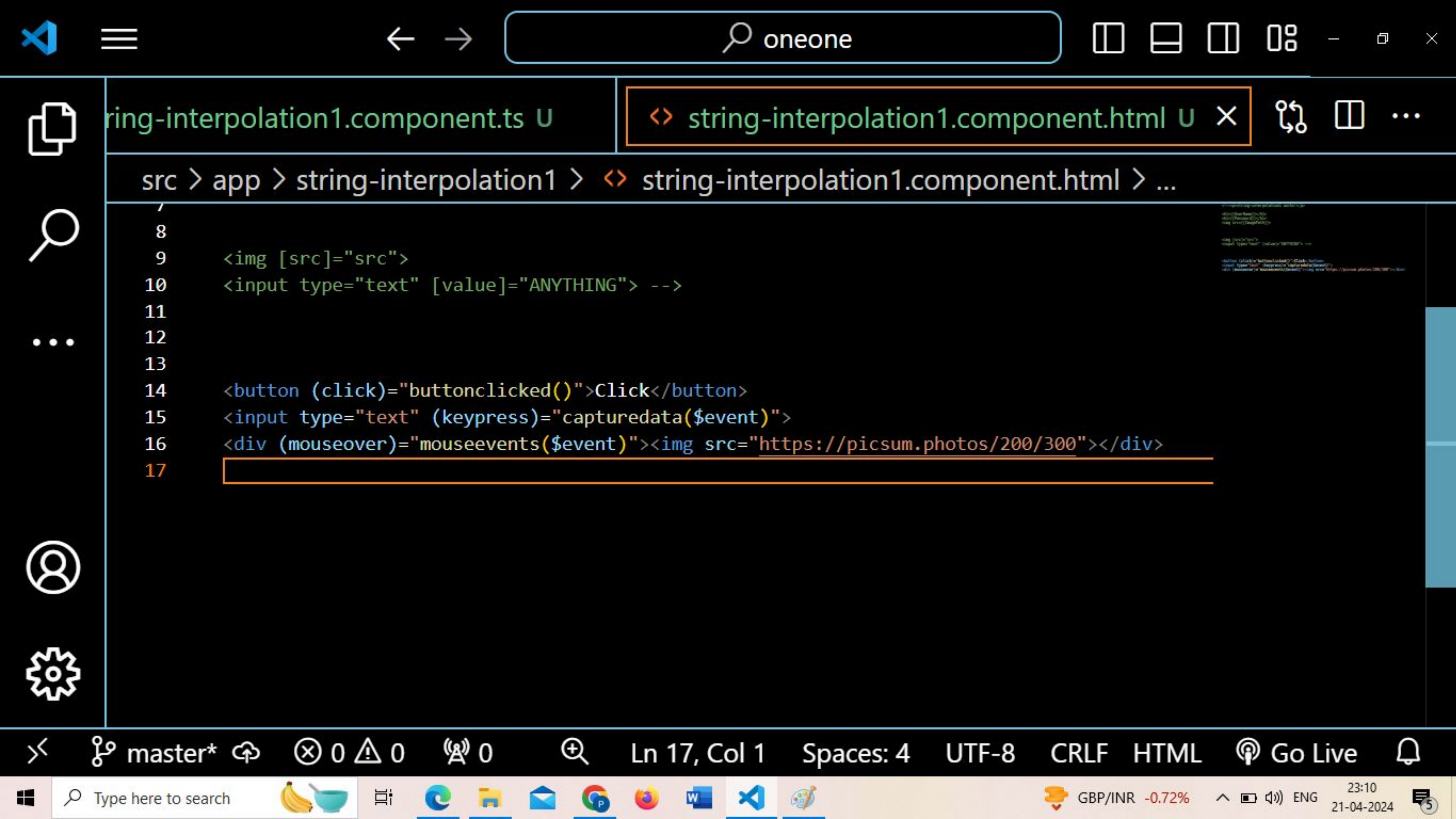
```
    imports: [],
    templateUrl: './string-interpolation1.component.html',
    styleUrl: './string-interpolation1.component.css'
  })
  export class StringInterpolation1Component {
  /*UserName:string="Palvi"
  Password:String="Soni"
  ImagePath:String="https://picsum.photos/200" */
  src="https://picsum.photos/300"
  ANYTHING="HIIIIII"
  }

```


localhost:56173

HIIIIII


oneone
TS string-interpolation1.component.ts U
string-interpolation1.component.htm
app > string-interpolation1 > TS string-interpolation1.component.ts > StringInterpolation1Compon
master*
0 0
0
Spaces: 2
UTF-8
CRLF
{} TypeScript
Go Live
Type here to search
GBP/INR -0.72%
ENG
23:10
21-04-2024

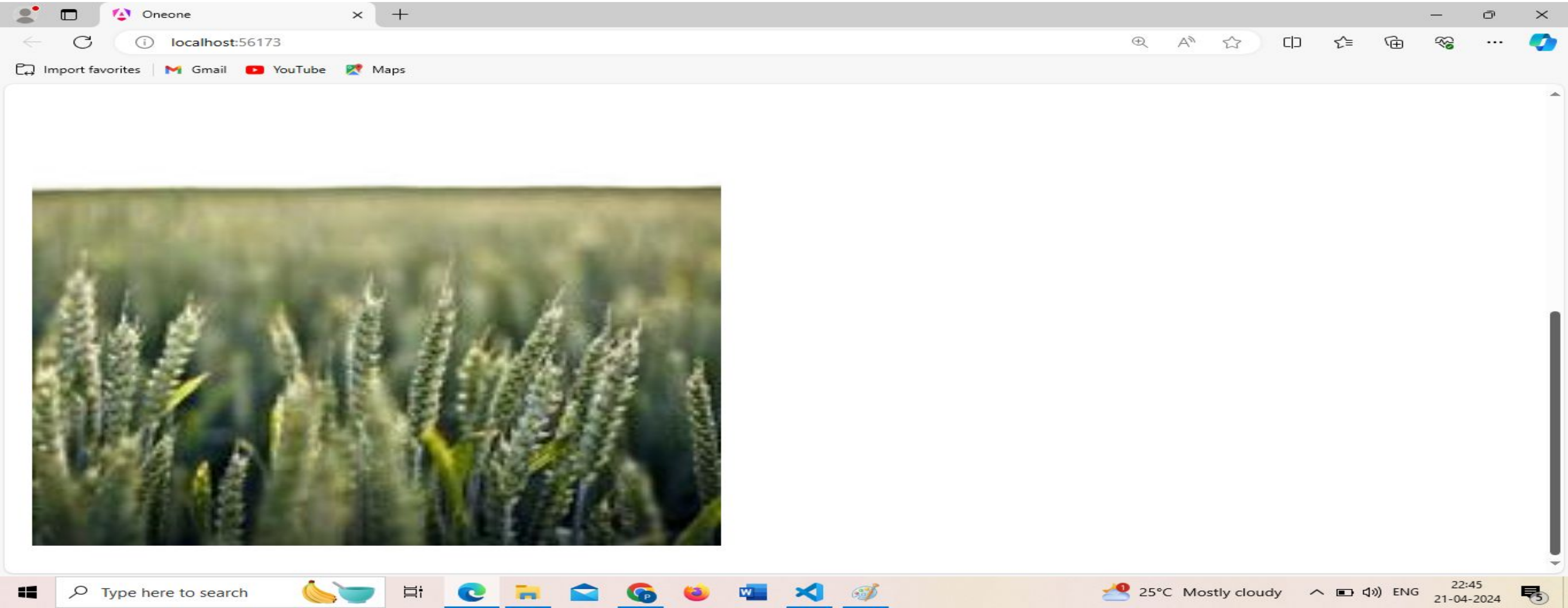
```
export class StringInterpolation1Component {
src="https://picsum.photos/300"
ANYTHING="HIIIIII" */
buttonclicked()
{
  document.write("You made a click on the button");
}
capturedata(event:any)
{
  document.write("keypress is"+" "+event.key)
}
mouseevents(event:any)
{
document.write("mouse is entered")
}
}
```



oneone
ring-interpolation1.component.ts U
string-interpolation1.component.html U
src > app > string-interpolation1 > string-interpolation1.component.html > ...

```
<img [src]="src">
<input type="text" [value]="ANYTHING"> -->



<button (click)="buttonclicked()">Click</button>
<input type="text" (keypress)="capturedata($event)">
<div (mouseover)="mouseevents($event)"><img src="https://picsum.photos/200/300"></div>

```

master* 0 0 0 Ln 17, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live
Type here to search
GBP/INR -0.72% ENG 23:10 21-04-2024


Oneone
localhost:56173
Import favorites
Gmail
YouTube
Maps
Type here to search
25°C Mostly cloudy
ENG
22:45
21-04-2024

Weekly Activity 12

Create a component with your name, and include two buttons, when you click on first button, the background color of button will changed to “brown”. When you click on the second button,your image will be displayed. Similarly when you make a click on that image, a new text in an input box will be opened, that displays a text “Lets Learn Angular”. Finally when you press a key inside the text box, the text will get vanished. Make use of event binding in your code.