

# 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` directive.

# 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)

The screenshot shows the Visual Studio Code (VS Code) editor interface. At the top, the search bar contains the text "oneone". Below the search bar, the file explorer on the left shows a project structure with "src > app > style1 > style1.component.html". The main editor area displays the content of "style1.component.html". The code is as follows:

```
1  <p>style1 works!</p>
2  <div [ngStyle]="{ 'color': 'red', 'font-size': '20px' }">
3    This text will be red and have a font size of 20px.
4  </div>
5
```

The status bar at the bottom indicates the current file is "master\*", the encoding is "UTF-8", the line endings are "CRLF", and the language is "HTML". The system tray at the bottom right shows the date and time as "21:52 21-04-2024".

# Import common module and compone

The screenshot shows the Visual Studio Code interface with the file explorer on the left and the editor in the center. The file explorer shows a project structure with 'src > app > style1' selected. The editor displays the file 'style1.component.ts' with the following code:

```
1 import { CommonModule } from '@angular/common';
2 import { Component } from '@angular/core';
3
4 @Component({
5   selector: 'app-style1',
6   standalone: true,
7   imports: [Style1Component, CommonModule],
8   templateUrl: './style1.component.html',
9   styleUrls: ['./style1.component.css']
10 })
11 export class Style1Component {
12
13
14 }
```

The code is written in TypeScript. The imports for `CommonModule` and `Component` are on lines 1 and 2. The `@Component` decorator is on line 4. The `imports` array on line 7 includes `Style1Component` and `CommonModule`. The `templateUrl` and `styleUrls` are specified on lines 8 and 9. The `export class Style1Component` is on line 11. The code is formatted with syntax highlighting.

# Open app.component.ts

The screenshot shows the Visual Studio Code editor interface. The top toolbar includes the VS Code logo, a hamburger menu, navigation arrows, a search bar containing 'oneone', and window management icons. Below the toolbar, the Explorer sidebar on the left shows a file tree with 'src > app > TS app.component.ts' selected. The main editor area displays the content of 'app.component.ts'. The file explorer tabs at the top show 'app.component.html' and 'TS app.component.ts' (the active file). The code in the editor is as follows:

```
1 import { Component } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3
4 import { CommonModule } from '@angular/common';
5 import { Style1Component } from './style1/style1.component';
6
7 @Component({
8   selector: 'app-root',
9   standalone: true,
10  imports: [RouterOutlet, CommonModule, Style1Component],
11  templateUrl: './app.component.html',
12  styleUrls: ['./app.component.css']
13 })
14 export class AppComponent {
```

Red rectangular boxes highlight the import statements (lines 1-5) and the imports array in the @Component decorator (line 10).




style1 works!

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

# Directive-NgIf

<> app.component.html M X

TS app.component.ts M

src > app > <> app.component.html >  div

Go to component

```
1 <div>
2 | <button (click)="toggleVisibility()">Toggle Visibility</button>
3 </div>
4
5 <div *ngIf="isVisible">
6 | <p></p>
7 </div>
8
9 <div *ngIf="!isVisible">
10 | <p>No Image</p>
11 </div>
```

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

```
3
4 import { CommonModule } from '@angular/common';
5
6 @Component({
7   selector: 'app-root',
8   standalone: true,
9   imports: [RouterOutlet, CommonModule],
10  templateUrl: './app.component.html',
11  styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   title = 'oneone';
15   isVisible: boolean = false;
16
17
18   toggleVisibility() {
19     this.isVisible = !this.isVisible;
20   }
21 }
22
```

# 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.

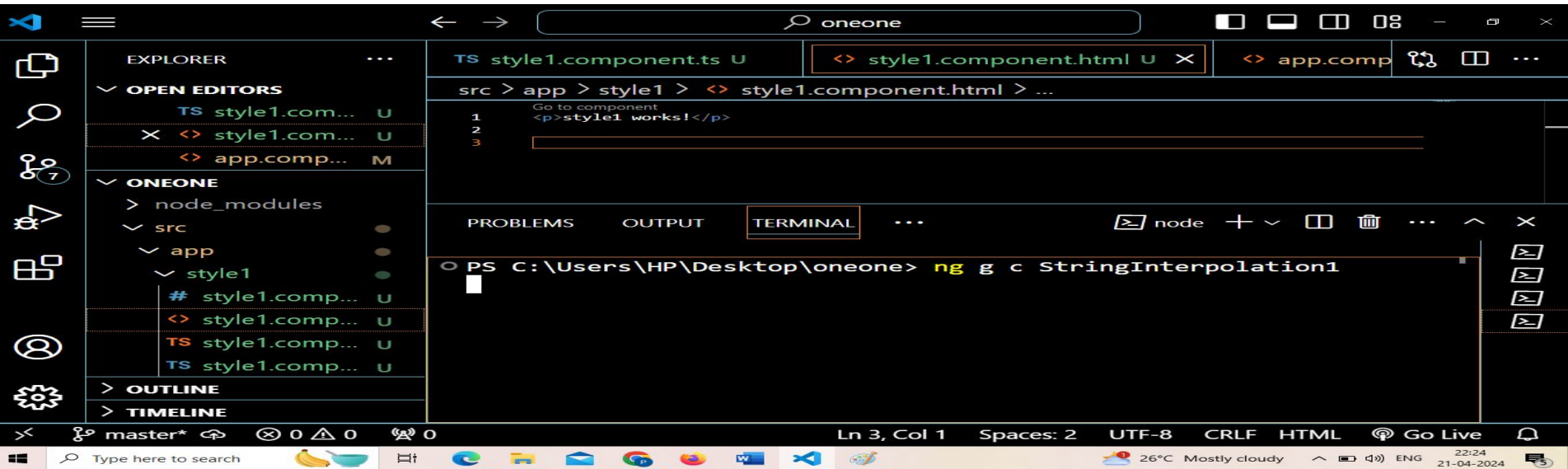
## 5. Template Reference Variables (`#var`):

- 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



# Open stringinterpolation.component.html

The screenshot shows the Visual Studio Code interface with the following components:

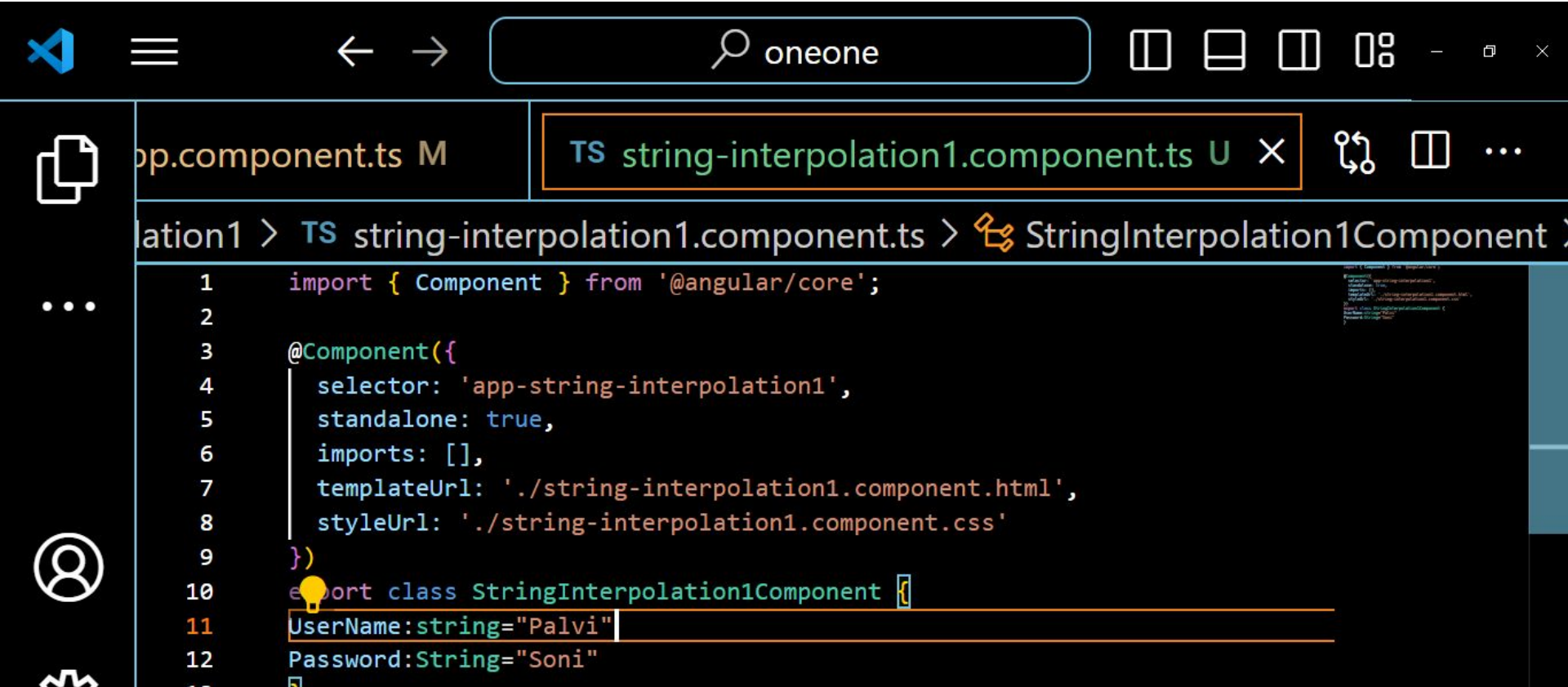
- Explorer View:** Displays the file structure under the **ONEONE** workspace. The **src** folder is expanded, showing the **app** folder, which contains the **string-interpolation1** component. The **string-interp...** file is selected.
- Open Editors View:** Shows the list of open files. The **string-inter...** file is the active editor.
- Editor View:** Displays the content of the **string-interpolation1.component.html** file. The file path is **src > app > string-interpolation1 > string-interpola**. The content is as follows:

```
1  Go to component
2  <p>string-interpolation1 works!</p>
3
4  <h1>{{UserName}}</h1>
   <h1>{{Password}}</h1>
```

The bottom status bar shows the current file is **master\***, with 0 errors, 0 warnings, and 0 info messages. The encoding is **UTF-8**, the line ending is **CRLF**, and the language is **HTML**. The **Go Live** button is also visible.



# Open stringinterpolation.component.ts



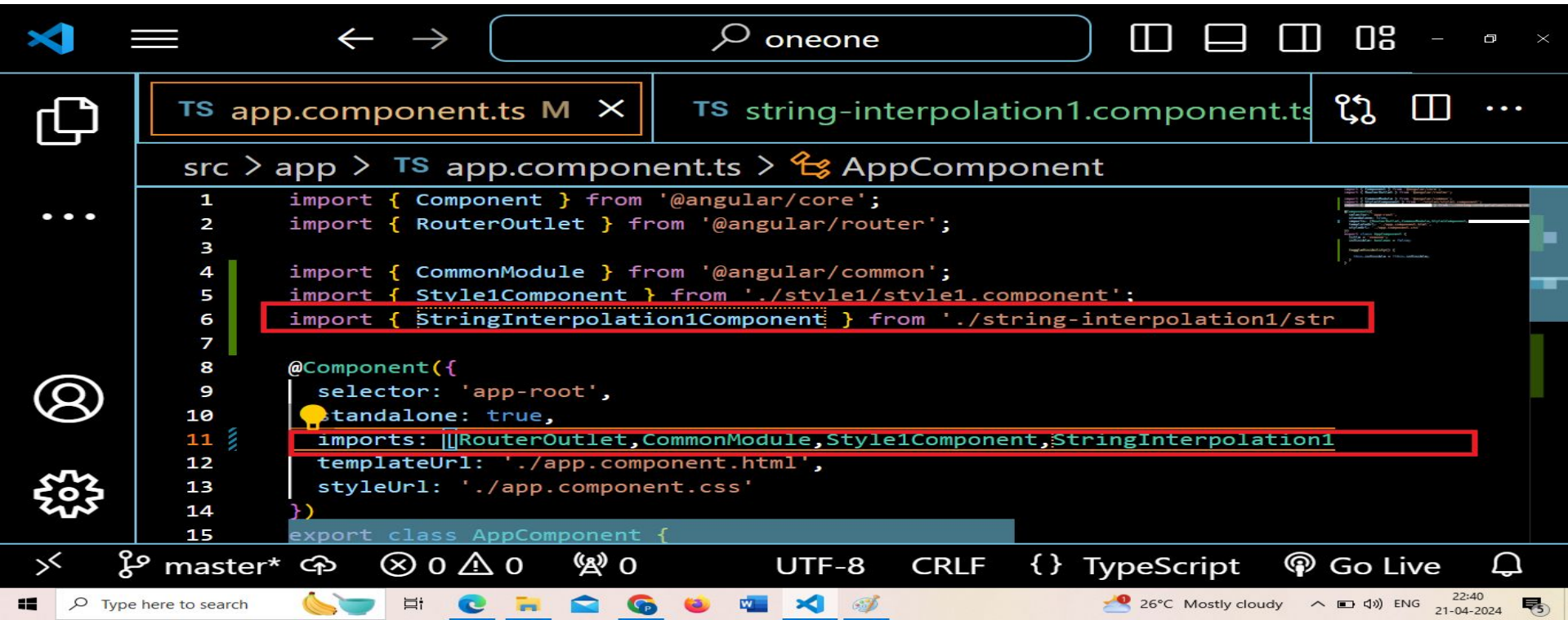


# Open app.component.html

The image shows a screenshot of the Visual Studio Code editor interface. The top bar includes the Visual Studio Code logo, a hamburger menu, navigation arrows, a search bar containing "oneone", and window management icons. The left sidebar contains icons for Explorer, Search, Source Control, and Settings. The Explorer view shows a file tree with "app.component.html" selected and highlighted with an orange border. The breadcrumb navigation shows the path: "src > app > app.component.html > app-string-interpolation1". The code editor displays the content of "app.component.html", which is a single line of HTML: `<app-string-interpolation1></app-string-interpolation1>`. The status bar at the bottom shows "master\*", "Spaces: 2", "UTF-8", "CRLF", "HTML", "Go Live", and a bell icon. The Windows taskbar at the very bottom shows the search bar, taskbar icons for various applications, and system information including temperature, weather, and time.

Visual Studio Code interface showing the file explorer, search bar, and code editor. The file explorer displays the file structure, including `app.component.html` (highlighted). The breadcrumb navigation shows the path: `src > app > app.component.html > app-string-interpolation1`. The code editor displays the content of `app.component.html`, which is a single line of HTML: `<app-string-interpolation1></app-string-interpolation1>`.

Open app.component.ts- import the necessary components



The screenshot shows the Visual Studio Code editor interface. The top toolbar includes the VS Code logo, a hamburger menu, navigation arrows, a search bar with the text 'oneone', and window management icons. The Explorer sidebar on the left shows a file icon and a settings gear. The Editor area displays two tabs: 'TS app.component.ts M' (active) and 'TS string-interpolation1.component.ts'. The active tab shows the following code:

```
src > app > TS app.component.ts > AppComponent
1  import { Component } from '@angular/core';
2  import { RouterOutlet } from '@angular/router';
3
4  import { CommonModule } from '@angular/common';
5  import { Style1Component } from './style1/style1.component';
6  import { StringInterpolation1Component } from './string-interpolation1/str
7
8  @Component({
9      selector: 'app-root',
10     standalone: true,
11     imports: [RouterOutlet, CommonModule, Style1Component, StringInterpolation1
12     templateUrl: './app.component.html',
13     styleUrls: ['./app.component.css']
14 })
15 export class AppComponent {
```

The imports on line 6 and the imports array in the decorator on line 11 are highlighted with red boxes. The bottom status bar shows 'master\*' with branch icons, 'UTF-8 CRLF' encoding, '{ } TypeScript' language, 'Go Live' button, and a bell icon. The Windows taskbar at the very bottom shows the search bar and several application icons.

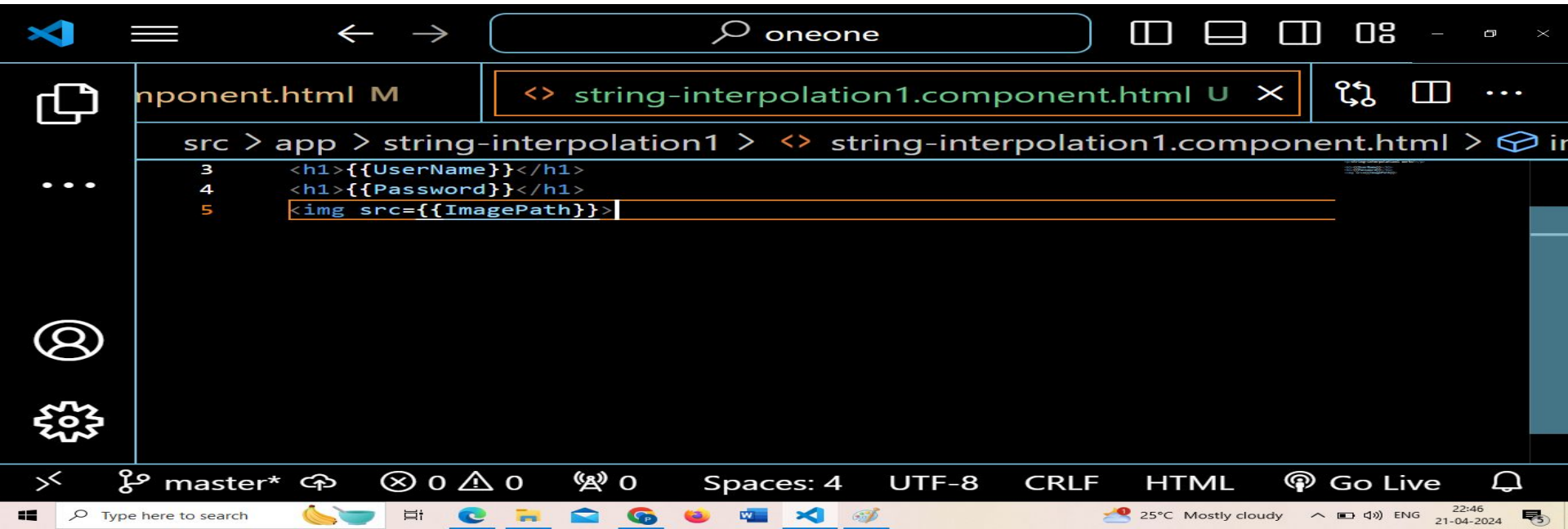
# output

The screenshot shows the Visual Studio Code editor interface. The top toolbar includes icons for Explorer, Search, Run and Debug, and a search bar containing 'oneone'. The Explorer sidebar on the left shows a file tree with 'app.component.ts' selected. The main editor area displays the content of 'app.component.ts' with the following code:

```
1 import { Component } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3
4 import { CommonModule } from '@angular/common';
5 import { Style1Component } from './style1/style1.component';
6 import { StringInterpolation1Component } from './string-interpolation1/str
7
8 @Component({
9   selector: 'app-root',
10  standalone: true,
11  imports: [RouterOutlet, CommonModule, Style1Component, StringInterpolation1
12  templateUrl: './app.component.html',
13  styleUrls: ['./app.component.css']
14 })
15 export class AppComponent {
```

Red boxes highlight the import statement on line 6 and the imports array in the @Component decorator on line 11. The status bar at the bottom shows 'master\*', '0 0 0', 'UTF-8', 'CRLF', '{} TypeScript', 'Go Live', and a system tray with the date '21-04-2024' and time '22:40'.

# String Interpolation for image



The screenshot shows a code editor interface with a dark theme. The top bar includes a search bar with the text "oneone" and several window management icons. The left sidebar shows a file explorer with a tree view containing the path "src > app > string-interpolation1 > string-interpolation1.component.html". The main editor area displays the following HTML code:

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

The code is written in a monospaced font. The file name "string-interpolation1.component.html" is highlighted in the top bar. The bottom status bar shows the file name "master\*", the encoding "UTF-8", the line ending "CRLF", the language "HTML", and the "Go Live" button. The system tray at the bottom right shows the temperature "25°C", the weather "Mostly cloudy", the time "22:46", and the date "21-04-2024".



oneone



TS string-interpolation1.component.ts U X

<> app.component.html



src > app > string-interpolation1 > TS string-interpolation1.component.ts > ...

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-string-interpolation1',
5   standalone: true,
6   imports: [],
7   templateUrl: './string-interpolation1.component.html',
8   styleUrls: ['./string-interpolation1.component.css']
9 })
10 export class StringInterpolation1Component {
11   UserName:string="Palvi"
12   Password:String="Soni"
13   ImagePath:String="https://picsum.photos/200"
14 }
15
```

TS string-interpolation1.component.ts



master\*



UTF-8

CRLF



TypeScript



Go Live



Type here to search



GBP/INR -0.72%



ENG

22:46

21-04-2024



# 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 X

TS



interpolation1 > <> string-interpolation1.component.html >  input

```
4    <h1>{{Password}}</h1>
5    <img src={{ImagePath}}>
6
7
8
9    <img [src]="src">
10   <input type="text" [value]="ANYTHING">
```



master\*



CRLF

HTML



Go Live



Type here to search



25°C Mostly cloudy



ENG

23:00

21-04-2024





oneone



TS string-interpolation1.component.ts U ●



src > app > string-interpolation1 > TS string-interpolation1.component.ts > ...

```
6   imports: [],
7   templateUrl: './string-interpolation1.component.html',
8   styleUrls: ['./string-interpolation1.component.css']
9 })
10 export class StringInterpolation1Component {
11   /*UserName:string="Palvi"
12   Password:String="Soni"
13   ImagePath:String="https://picsum.photos/200" */
14   src="https://picsum.photos/300"
15   ANYTHING="HIIIIII"
16 }
17
```

```
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';

@Component({
  selector: 'app-string-interpolation1',
  templateUrl: './string-interpolation1.component.html',
  styleUrls: ['./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"
}
```



master\*



0

UTF-8

CRLF



TypeScript



Go Live



Type here to search



GBP/INR -0.72%



ENG

23:01

21-04-2024







HIIIII



oneone



TS string-interpolation1.component.ts U X

<> string-interpolation1.component.htm



c > app > string-interpolation1 > TS string-interpolation1.component.ts > StringInterpolation1Compon

```
10 export class StringInterpolation1Component {
14   src="https://picsum.photos/300"
15   ANYTHING="HIIIIIII" */
16   buttonclicked()
17   {
18     | document.write("You made a click on the button");
19   }
20   capturedata(event:any)
21   {
22     | document.write("keypress is"+" "+event.key)
23   }
24   mouseevents(_event:any)
25   {
26     | document.write("mouse is entered")
27   }
28 }
29
```

```
StringInterpolation1Component.ts
10 export class StringInterpolation1Component {
11   src="https://picsum.photos/300"
12   ANYTHING="HIIIIIII" */
13   buttonclicked()
14   {
15     document.write("You made a click on the button");
16   }
17   capturedata(event:any)
18   {
19     document.write("keypress is"+" "+event.key)
20   }
21   mouseevents(_event:any)
22   {
23     document.write("mouse is entered")
24   }
25 }
26
```



master\*



Spaces: 2

UTF-8

CRLF



TypeScript



Go Live



Type here to search



GBP/INR -0.72%

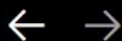


ENG

23:10

21-04-2024





oneone



string-interpolation1.component.ts U

<> string-interpolation1.component.html U X



src > app > string-interpolation1 > <> string-interpolation1.component.html > ...

```
7
8
9   <img [src]="src">
10  <input type="text" [value]="ANYTHING"> -->
11
12
13
14  <button (click)="buttonclicked()">Click</button>
15  <input type="text" (keypress)="capturedata($event)">
16  <div (mouseover)="mouseevents($event)"></div>
17
```

string-interpolation1.component.html



master\*



Ln 17, Col 1

Spaces: 4

UTF-8

CRLF

HTML



Go Live



Type here to search



GBP/INR -0.72%

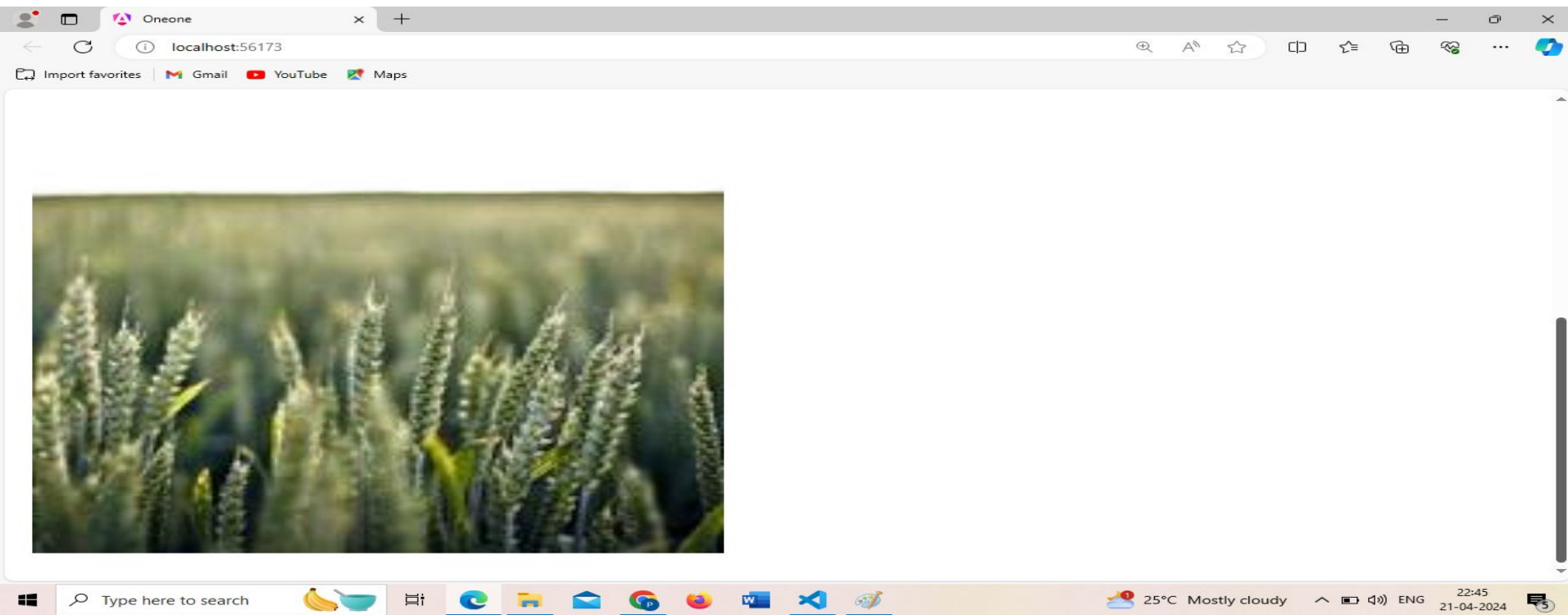


ENG

23:10

21-04-2024





## 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.