**Angular 19 Preparation – Notes from Learning Partner Videos**

**VIDEO: 2\_GMT20250411-014800\_Recording\_1920x1080**

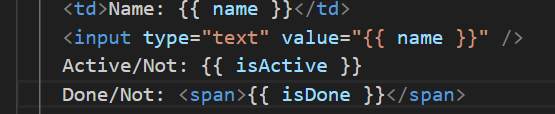
**What is Data Binding?--> Display a property, variable, or function value from the .ts file in the HTML UI, you can use string interpolation, property binding, or event binding.**

There are four main types of data binding in Angular:

**One-way binding**

1. **Interpolation** - {{}} → One-way binding from component to view.

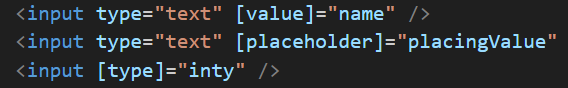
We use interpolation to display variables defined in the .ts file within the .html template.

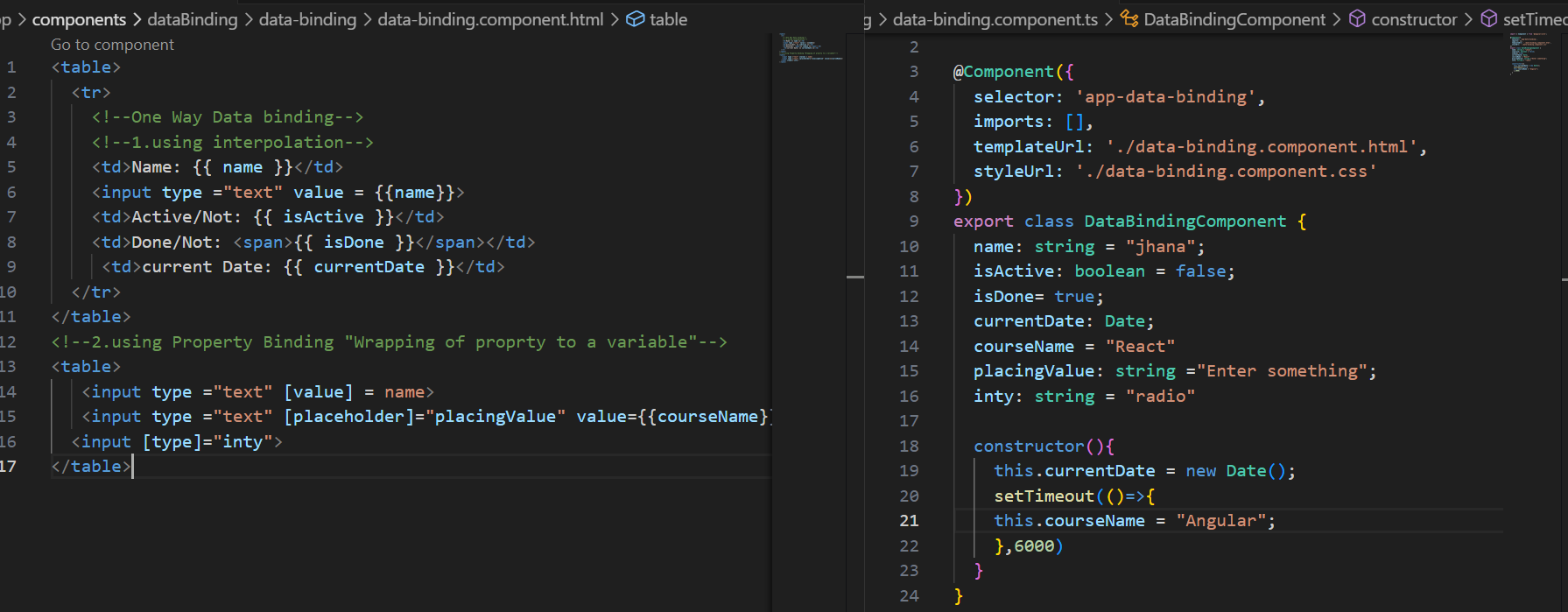


1. **Property Binding** - [property] → One-way binding from component to an HTML element.

Element properties can be made dynamic in Angular using property binding.

**To make a property dynamic, we wrap it in square brackets and pass only the variable name inside.**

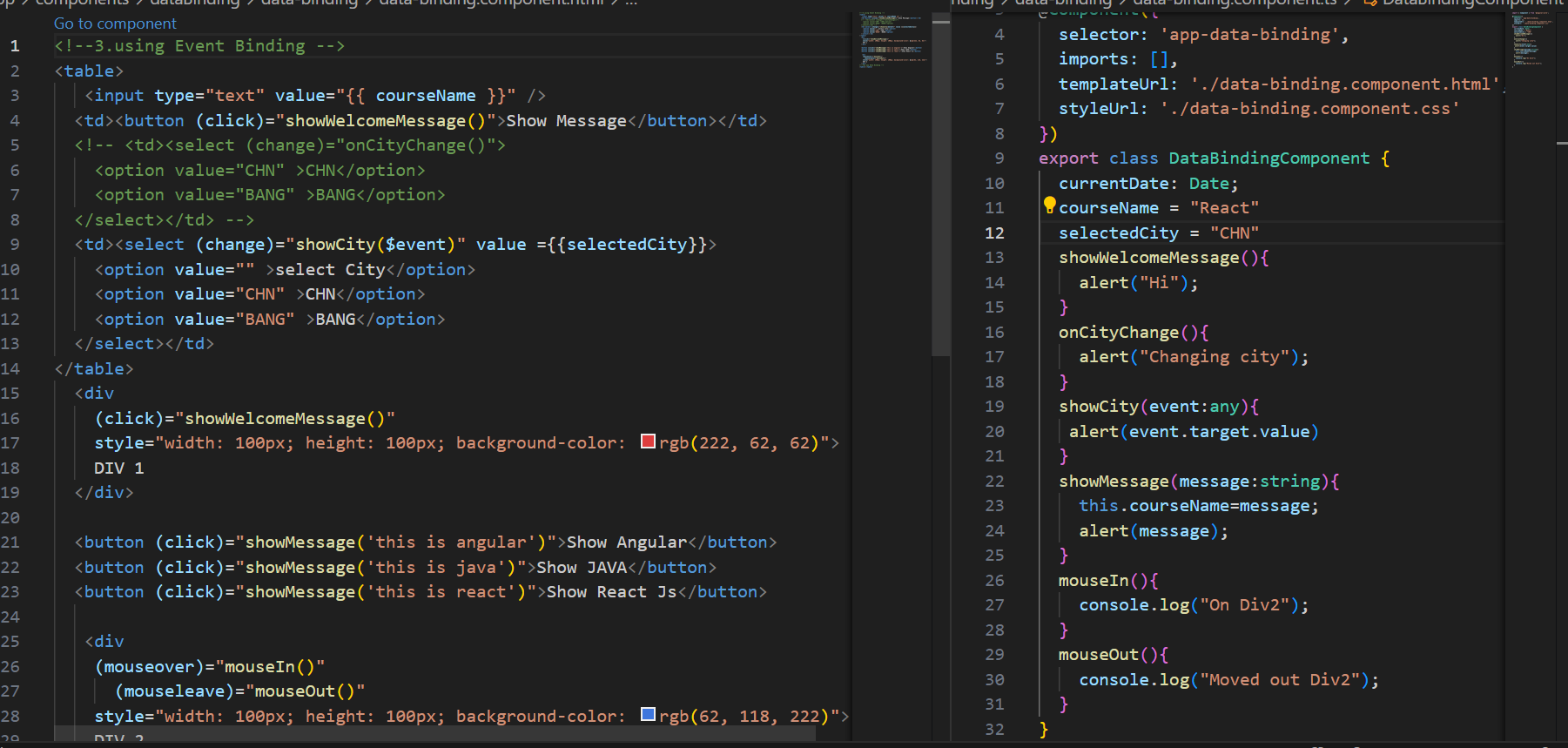


****

1. **Event Binding** - (event) → One-way binding from UI to the component.

**(event)=””🡪sending value from .html to .ts file**

**Hint: The change event is triggered only after the user clicks outside the textbox or presses Enter.**

****

**VIDEO: 3\_GMT20250414-014216\_Recording\_1920x1080**

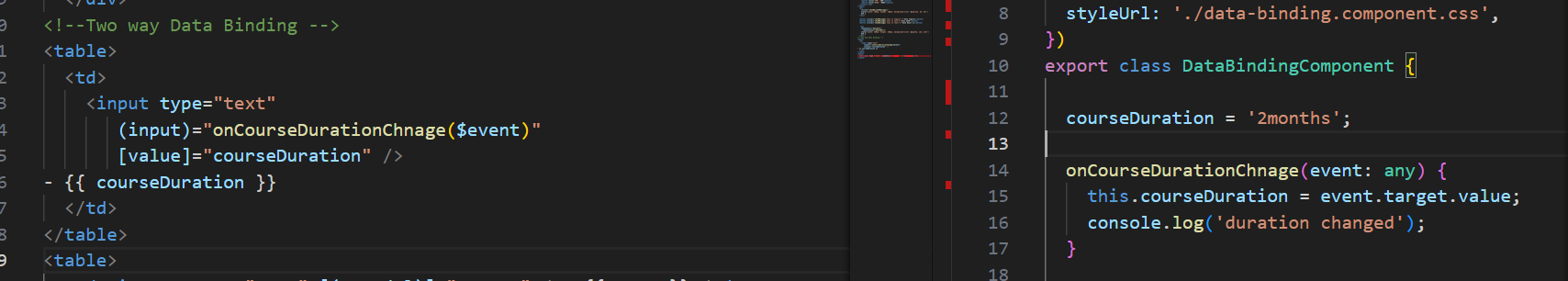
**Two-way binding**

1. **Two-Way Binding** - [(ngModel)] → Two-way sync between component and UI.

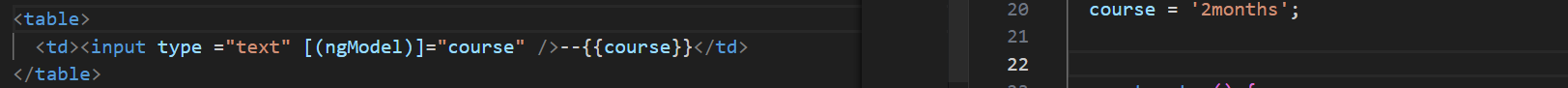
Whenever the user changes the value in the textbox, it should be captured and assigned to a variable.

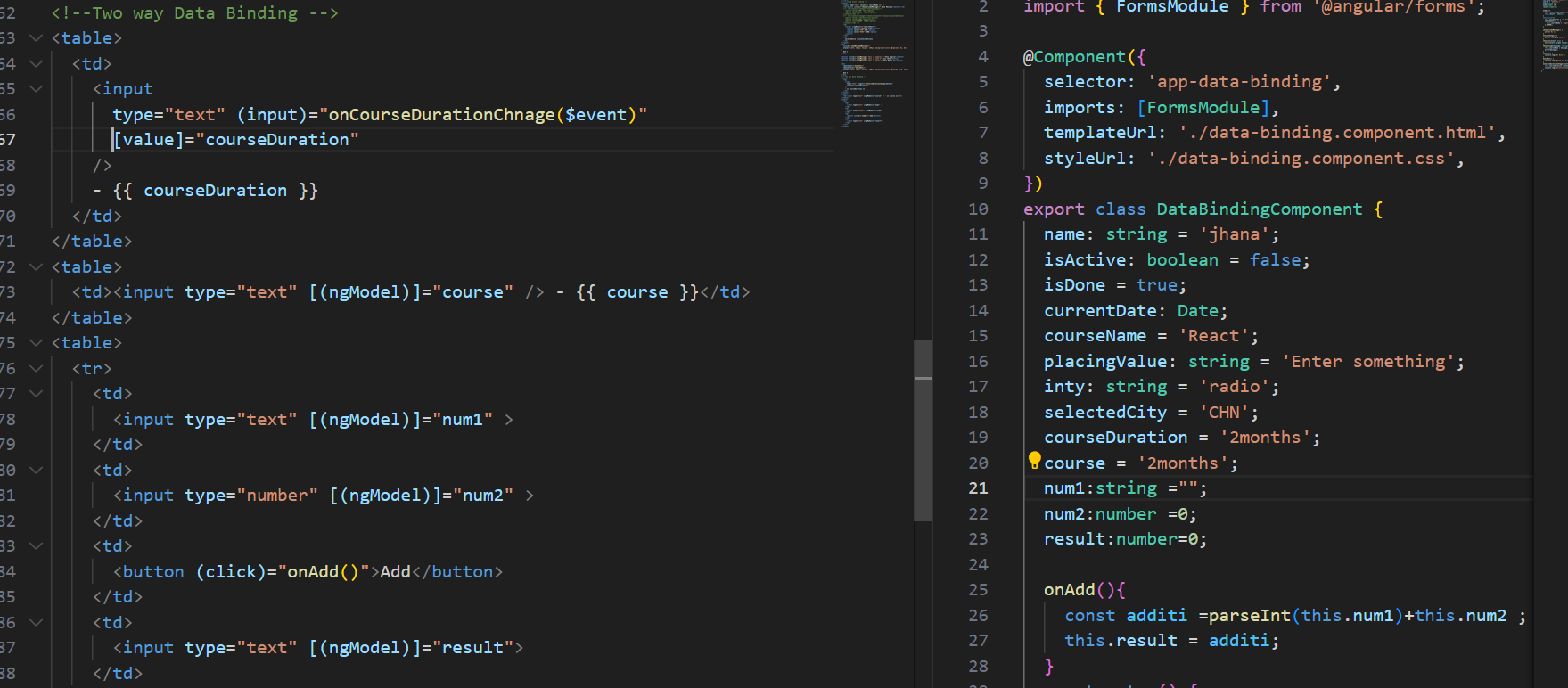
**The input, keyup, and keydown events are triggered when the user types. However, using the input event is a better and simpler approach compared to keyup or keydown.**

**We can achieve two-way data binding using the input event by creating a function and accessing event.target.value. However, instead of writing multiple lines of code, we can simply use ngModel in Angular.**

****

**ngModel Directive**

****

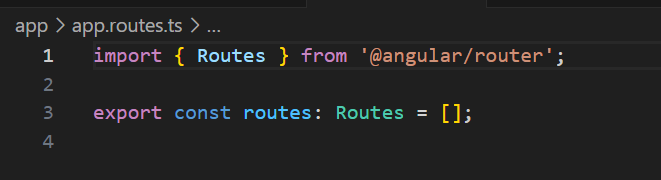
****

**Routes:**

**To achieve page-based navigation in Angular, we use the Angular Router library.**

**In angular 15 to 17 which is app-routing.module.ts**

**From version 18 we have app.routes.ts is generated when you create a new Angular project.**

****

It is just a constant file where we define the required navigation routes inside an array[].

**Routing is an array of objects. Each object should contain at least two properties: path and component.**

* path: The URL path for the component.
* component: The component to be rendered for that path.

We need to route all components except the parent (app.component) component.

In Angular, using href will cause a full page reload, which breaks the Single Page Application (SPA) behavior.

To enable proper navigation in Angular without reloading the page, you must use the routerLink directive instead of href.

**RouterLink must be imported in app.component.ts in order to use it.**

**router-outlet is used to display the routed component.**

The **router-outlet directive is required to render the component associated with the activated route**. It must be added to app.component.html.

**RouterOutlet must also be imported in app.component.ts in order to use it.**

**VIDEO: 4\_GMT20250415-014417\_Recording\_1920x1080**

**What is Directive—>perform a particular task**

Directives are of two types:

**1. Structural Directives –** Modify the DOM structure

* \*ngIf: Used to show or hide an element. (@if)

Conditionally includes or excludes an element from the DOM

* \*ngFor: Responsible for creating dynamic elements. (@for)

Iterates over a collection to create multiple elements dynamically.

* \*ngSwitch: Used to switch between multiple views based on a condition.

Switches among multiple views based on a condition.

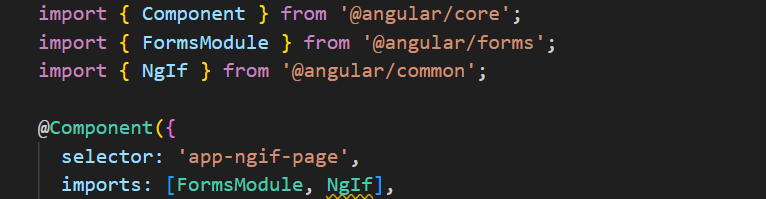
**2. Attribute Directives –** Modify the appearance or behavior of an element

* [ngClass]: Dynamically add or remove CSS classes.
* [ngStyle]: Apply inline CSS styles conditionally.

To use **ngModel**, you must import the **FormsModule** in the module and include it in the imports array:

To show or hide elements, you can use **\*ngIf**. For that, you must import the **CommonModule** in the module file and include it in the imports array.

It needs **true or false** value (**boolean value**)

****

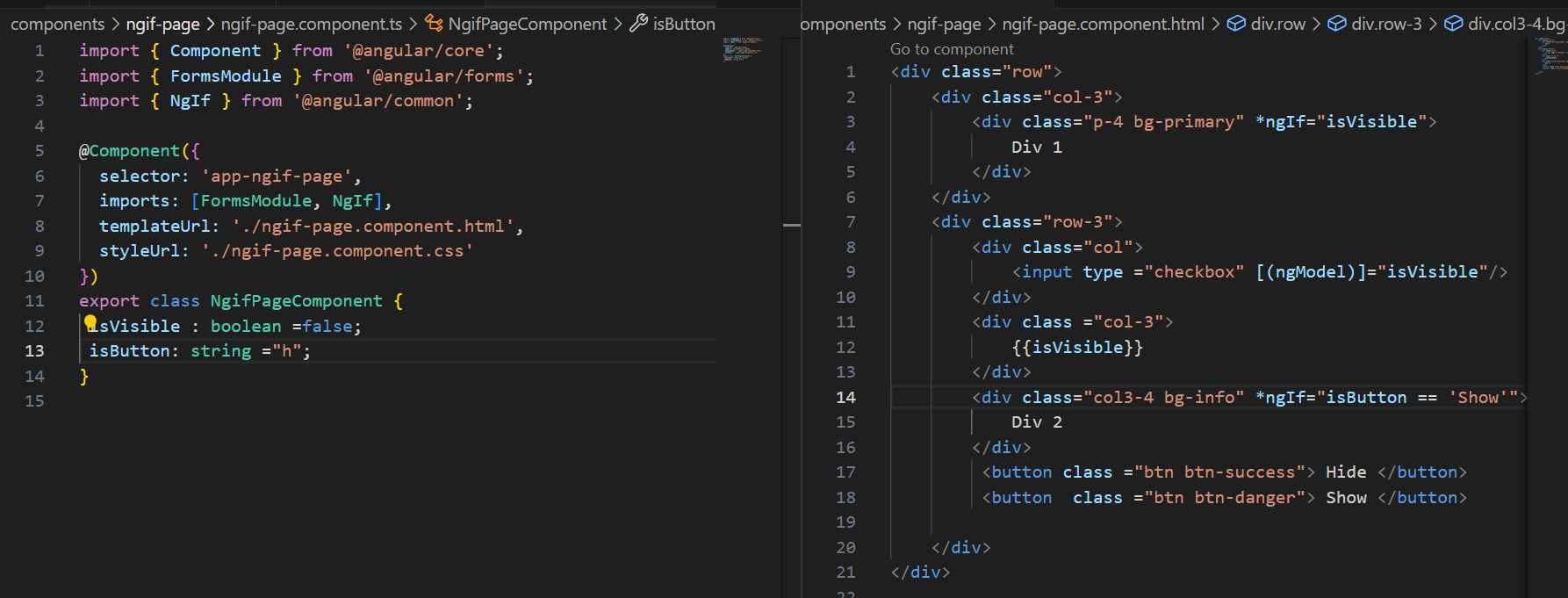
**If we use** \*ngIf, **and the condition is satisfied, Angular** adds the element to the DOM**. If the condition is not satisfied, it** removes the element from the DOM**.**

**1: Show a <div> if the value is true, hide it otherwise.**

**2: Toggle <div> visibility on button click.**

**3.** **We have radio buttons to select an option. If the user selects 'Other', a textbox will appear to enter a custom value.**

**A static value can be used as shown in the example below.**

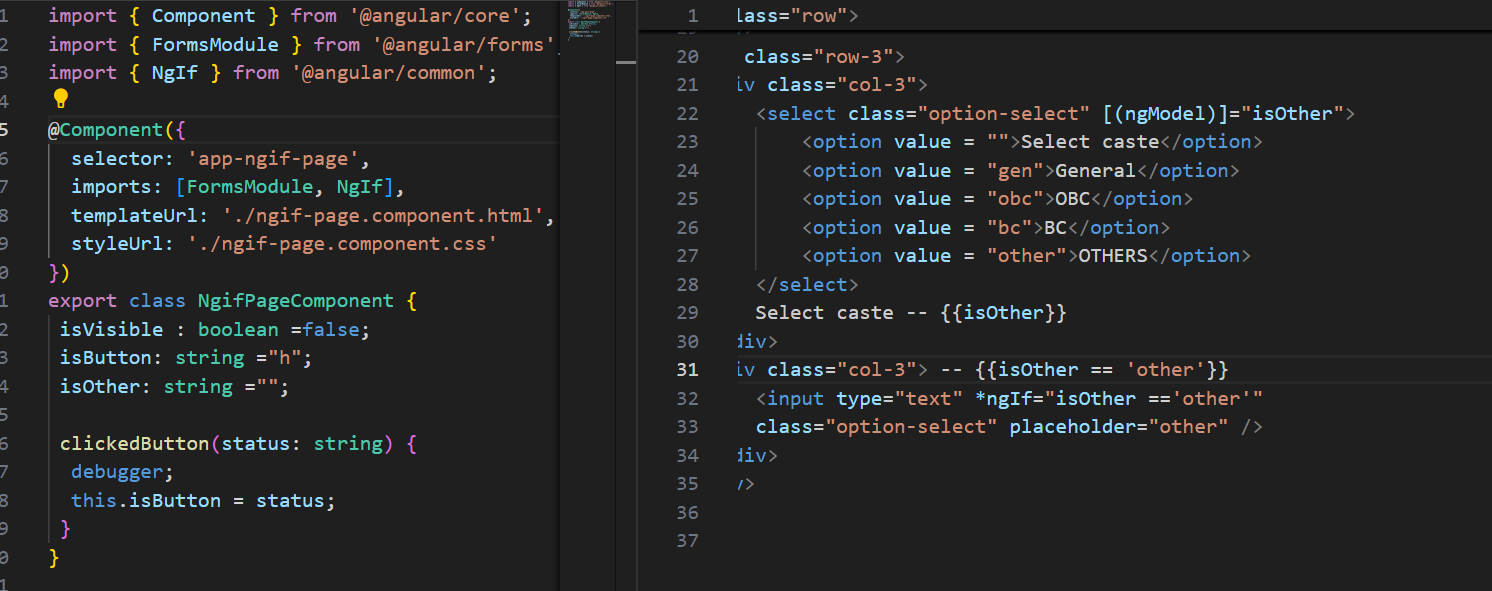
By creating a variable with a static value, if the condition is satisfied, the element will be shown; otherwise, it will be hidden.

**Dynamic behaviour with example.**

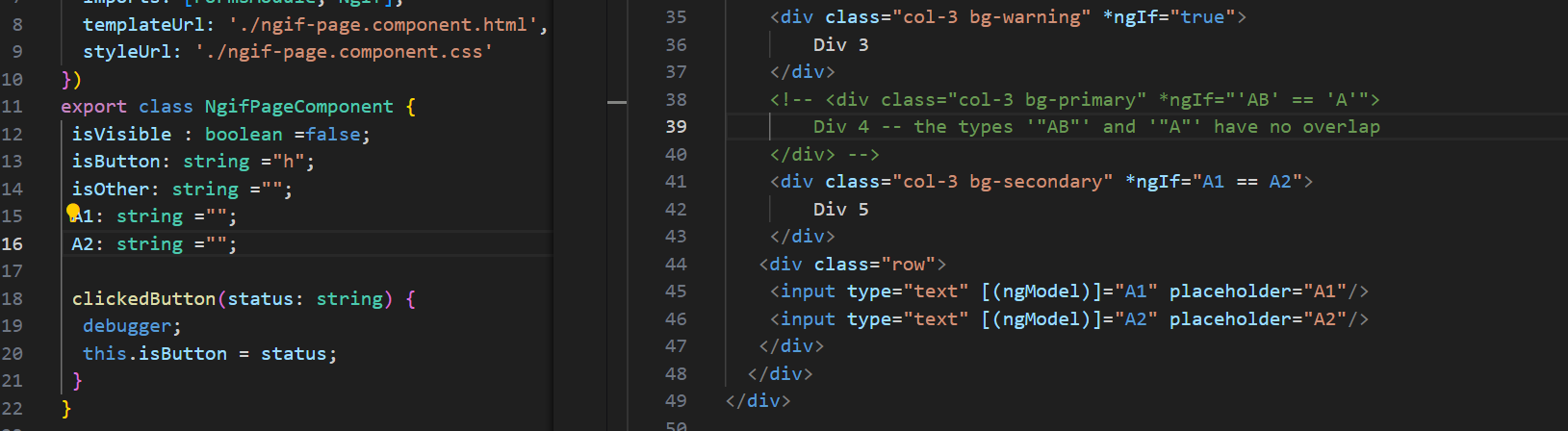
****

****

**3. We have radio buttons to select an option. If the user selects 'Other', a textbox will appear to enter a custom value.**

****

**Inline Boolean Logic (No Component Variable Needed**

****

**Another example:**

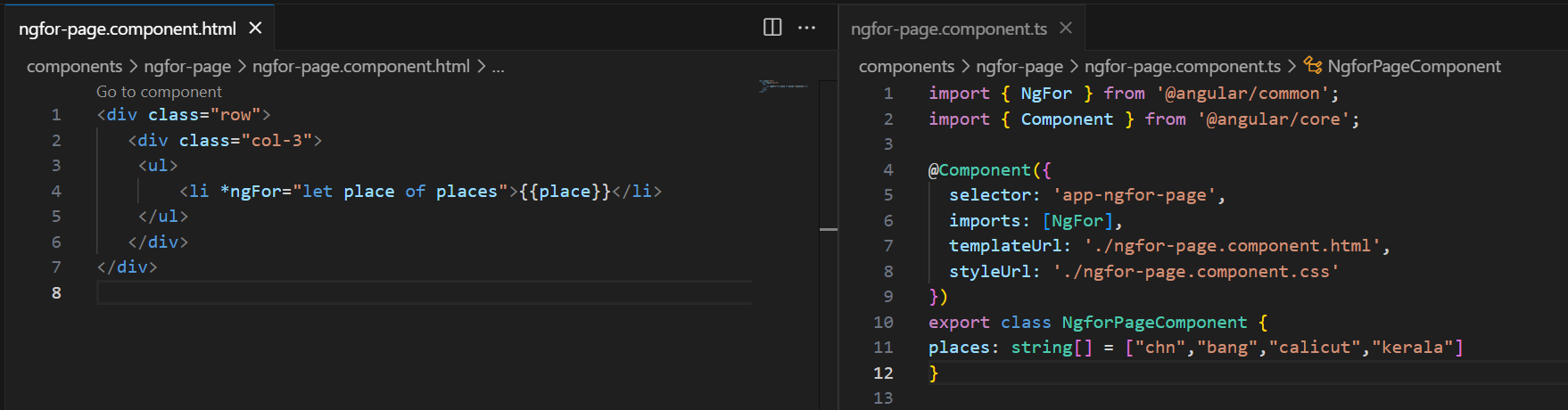
**The input fields should be mandatory to ensure we can calculate the result. Based on the values entered, when the Result button is clicked, if the total is an even number, it should display 'Even'; otherwise, it should display 'Odd'..**

****

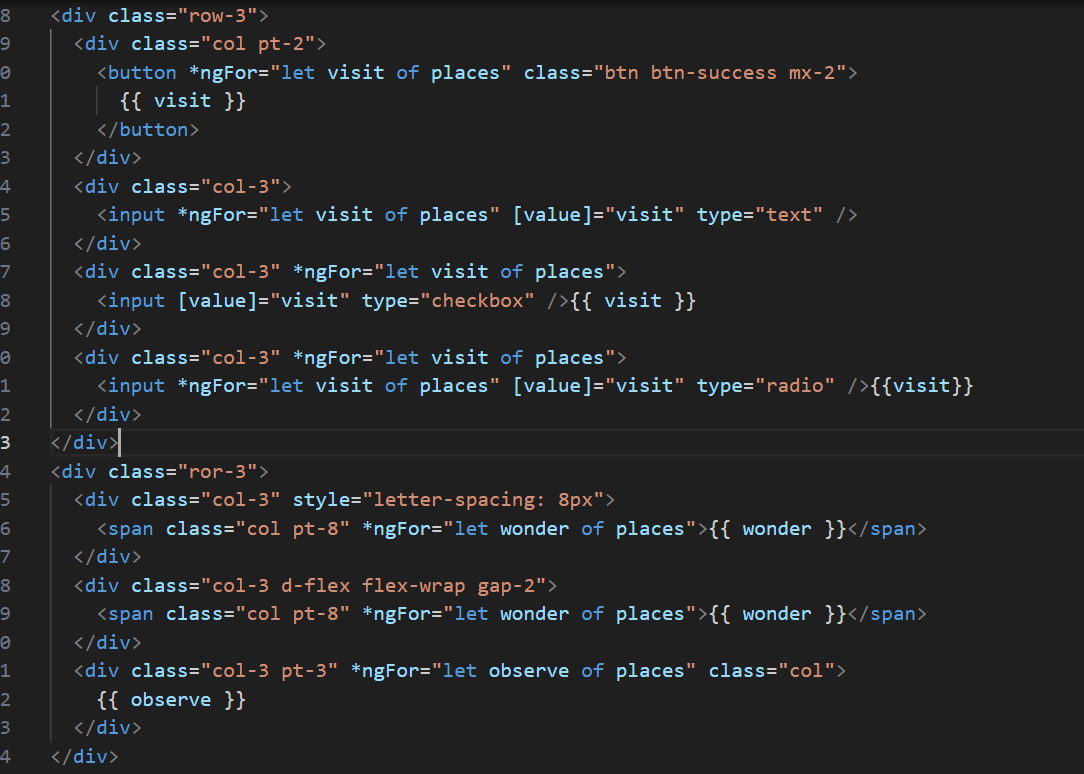
**VIDEO: 5\_Angular Batch -29 \_ Recording 16th Apr**

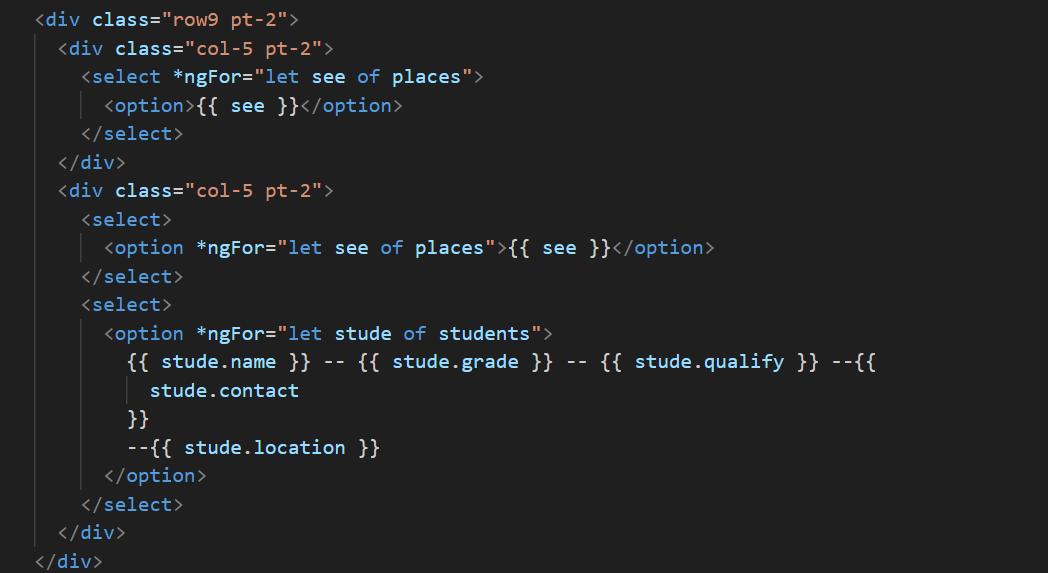
In Angular, to use **\*ngFor,** we need only an array.

ngFor: Creating a new element according to the array.

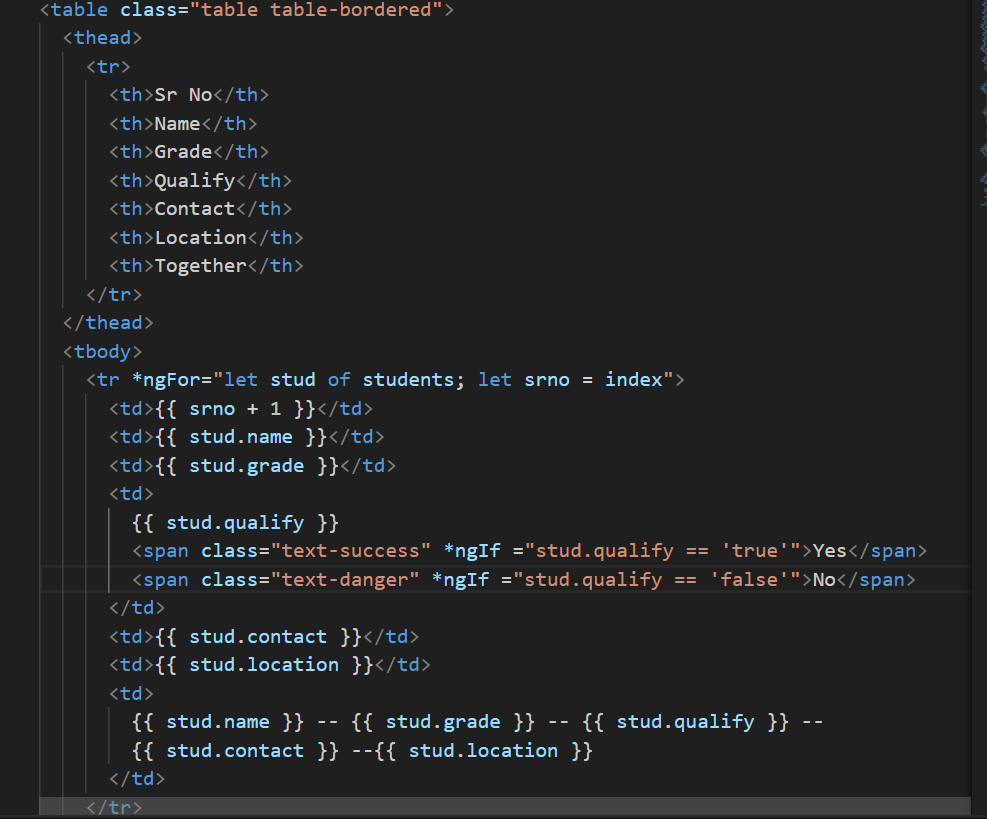


When you use \*ngFor, the variable you declare is accessible only within the element where \*ngFor is applied.





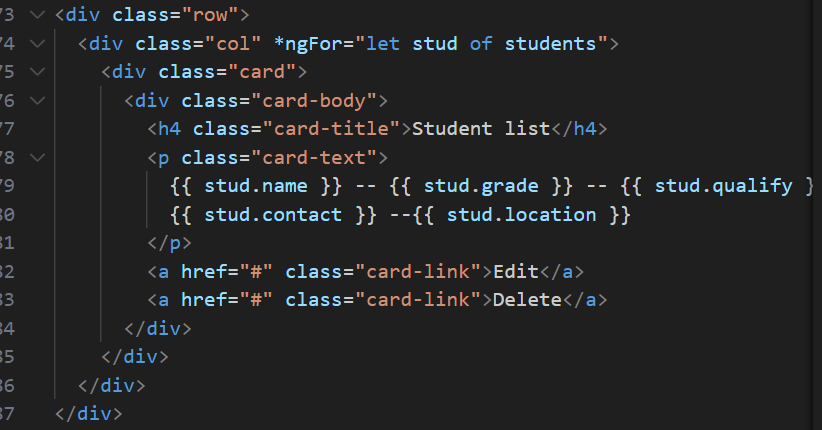
Angular's \*ngFor directive offers special local variables such **as index, first, last, even, and odd** to help you manage iterations more effectively.

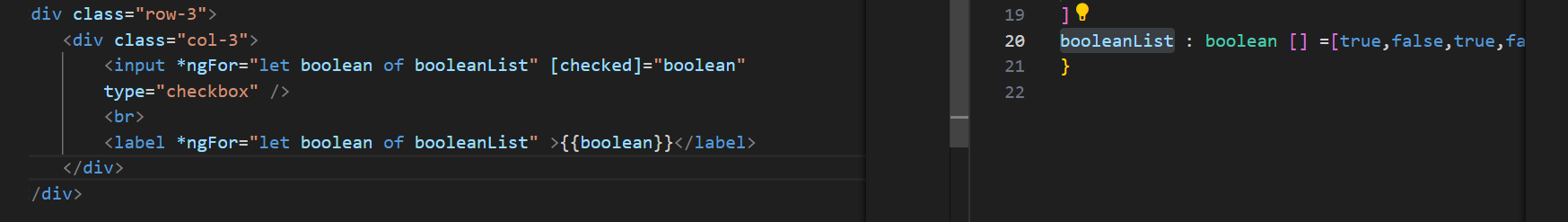


When using the index property in **\*ngFor, it starts from 0 by default.** To display a serial number starting from 1, you can add 1 to the index like this.



**Another way to use \*ngFor is by displaying data using Bootstrap cards.**



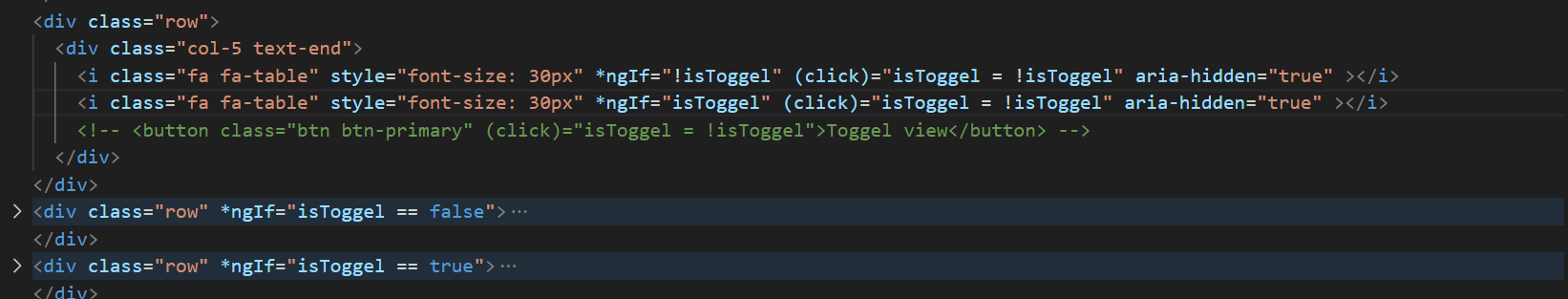


When to use a card and when to use a table.

Card: limited information – user friendly or trendy

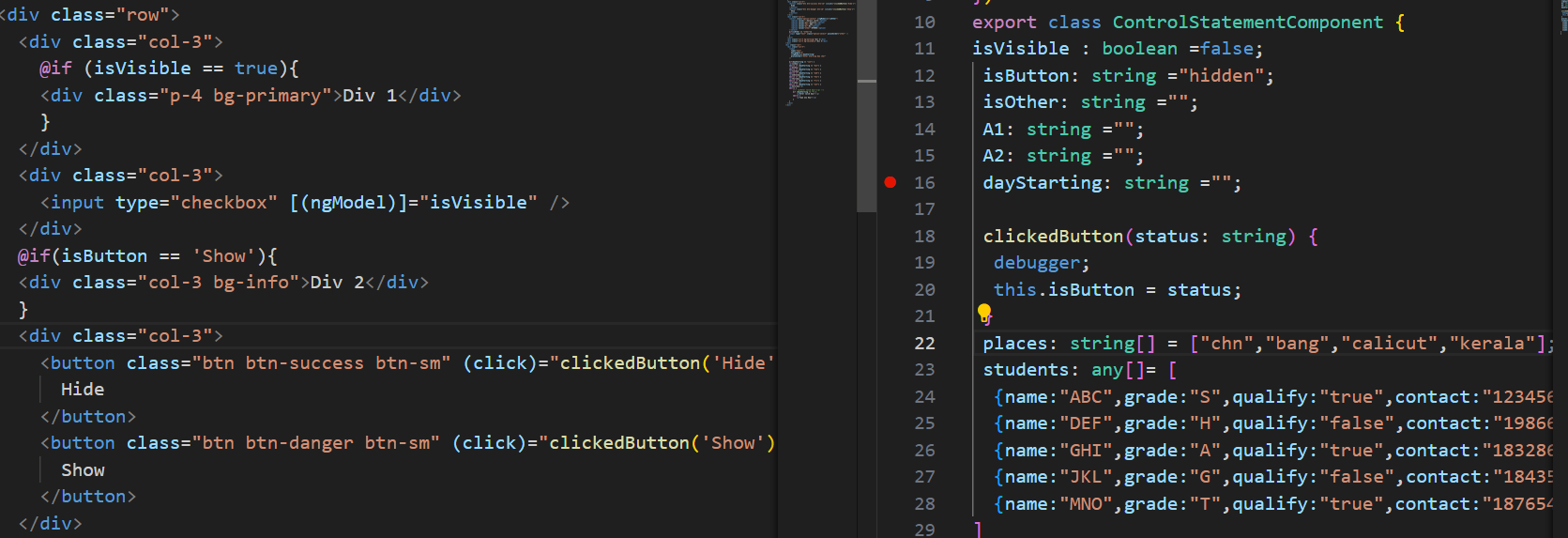
Table: More information

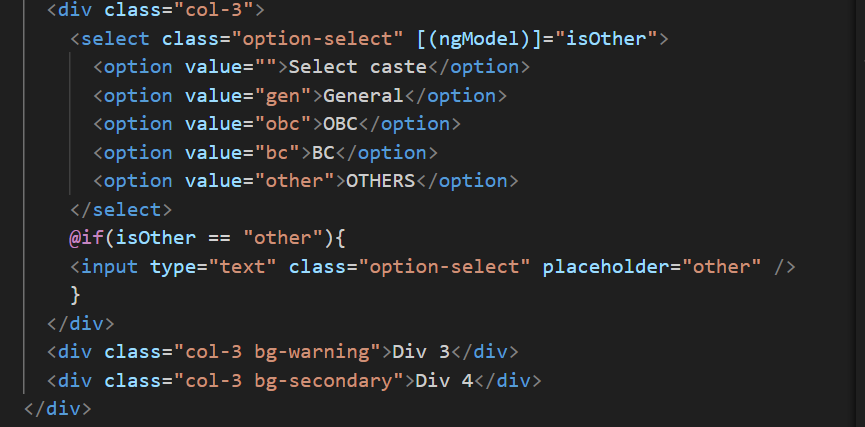
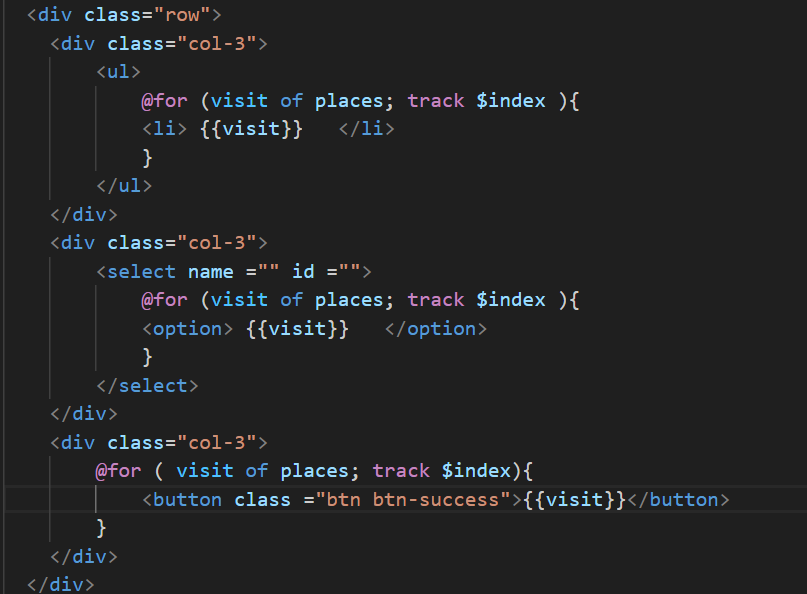
If the UI is showing the table view, the grid icon should be visible, and vice versa.

Code of STATIC view**VIDEO: 6\_Angular Batch -29 \_ Recording 17th Apr**

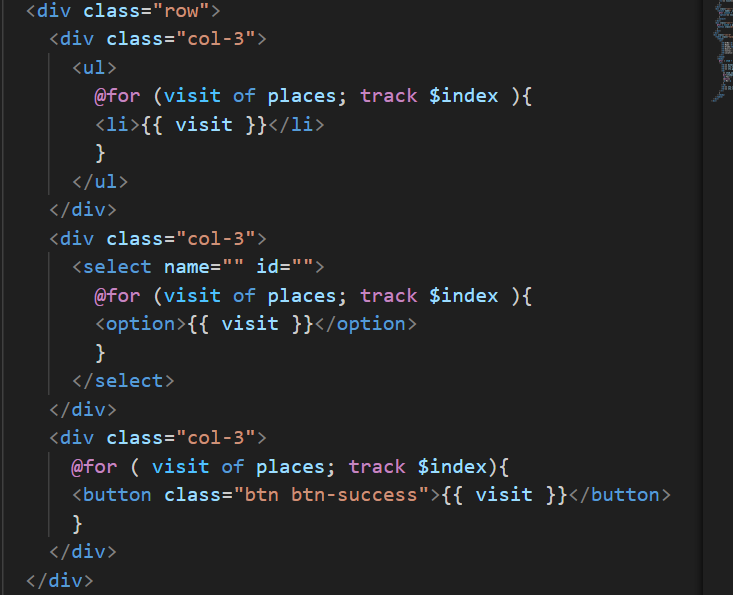
**Control flow statement** introduced from Angular 17

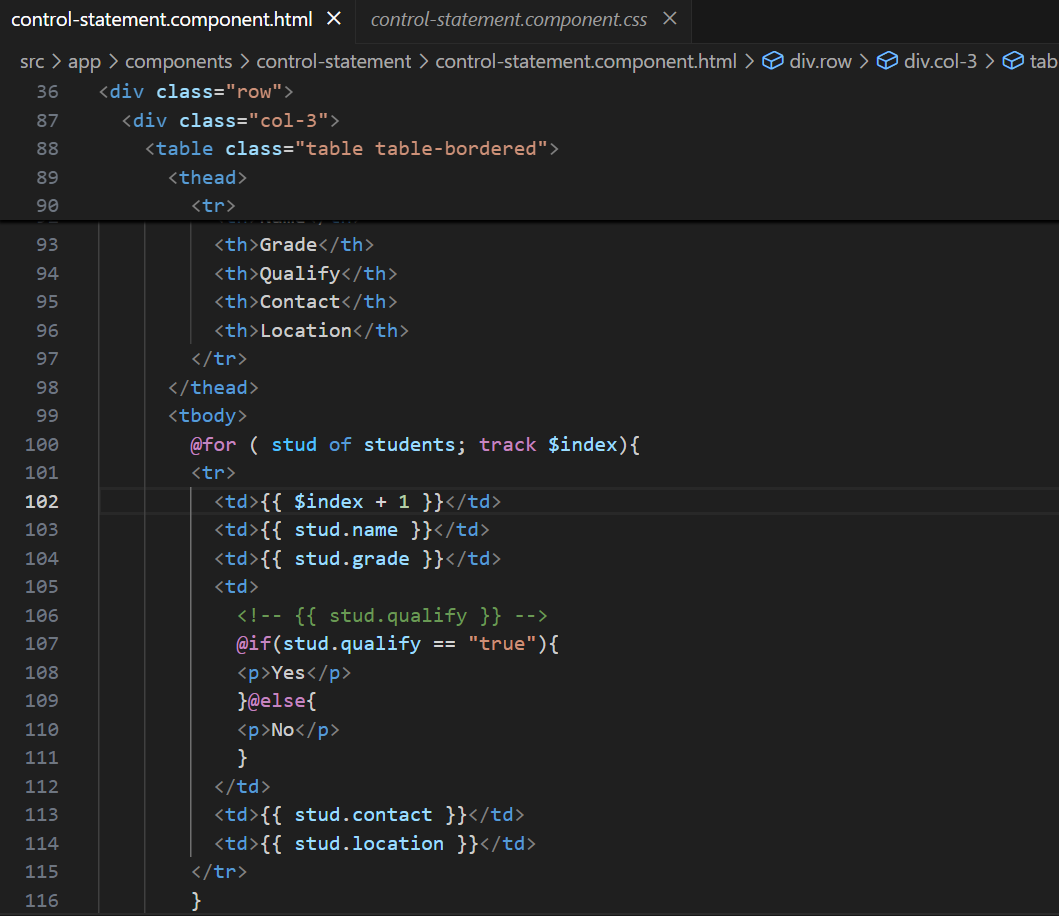
Import: formsModule @if and @for is less time complexity.







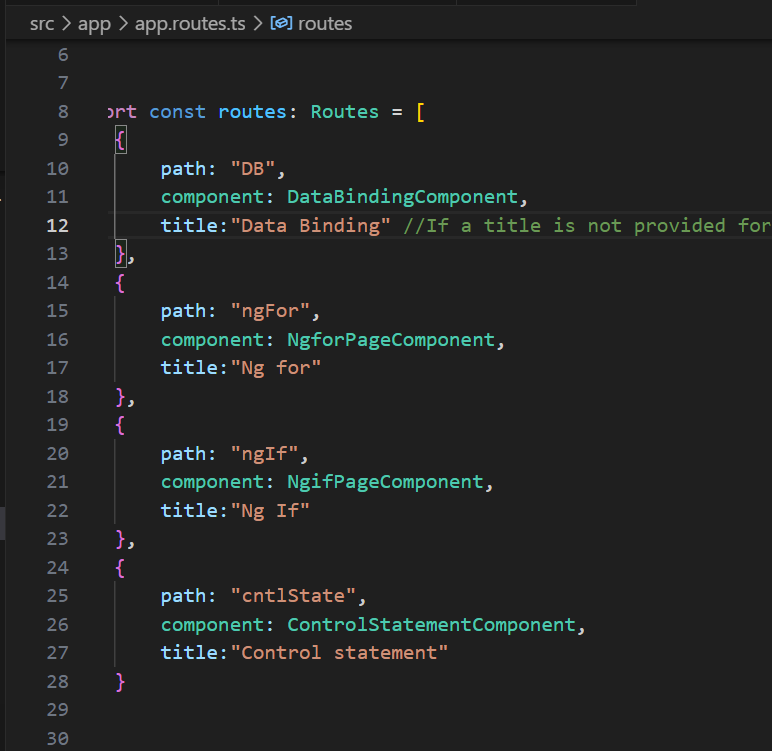




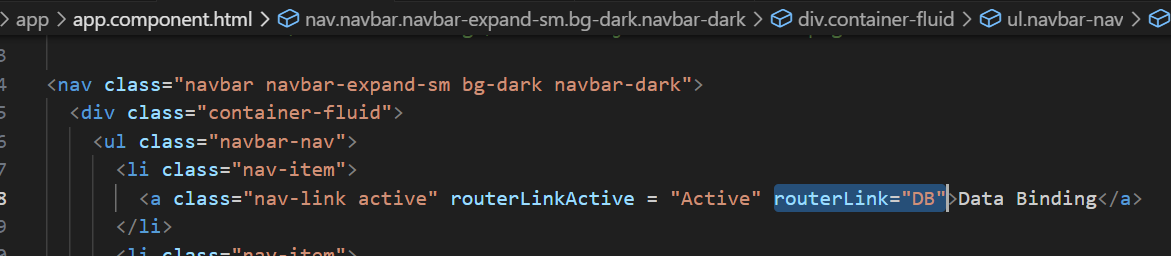
Highlight the currently active navigation link (e.g., add an Active class when a link matches the current route).

First step: **don't need to import RouterLinkActive** in app.component.ts.

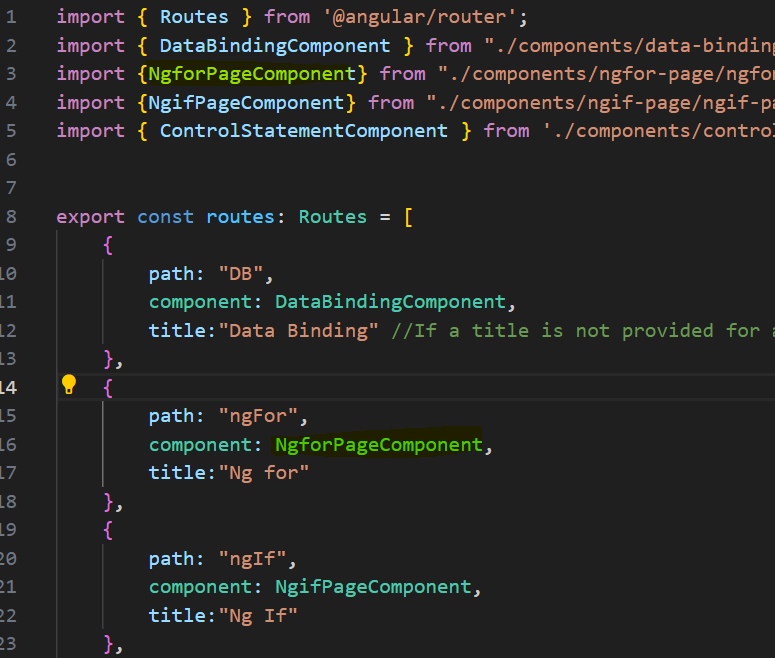
use routerLinkActive = "Active" in app.component.html



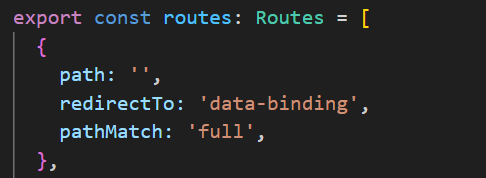
**Path** defines where the app should navigate. It should match the route defined in app.component.html using routerLink.

Example:  


Component: component name



If a title is not provided for a route, the title defined in index.html will be used as the default for all routes.



**path: ""** when the URL path is empty “root URL, like http://localhost:4200/.”

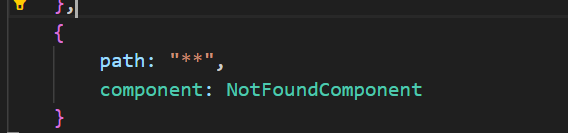
**🔹 redirectTo: "data-binding"**

* This tells Angular to **redirect** from the default path to /data-binding.

**🔹 pathMatch: "full"**

* Ensures that the **entire URL** must match the empty string ("") for the redirect to happen.
* Without "full", it could match partially and cause **unexpected behavior.**

To avoid unexpected behavior, we should **create a component** to handle unknown routes, which will be displayed when the user types an invalid or non-existent URL.



**Interview preparation link**

[**https://learningpartner.github.io/javaScript\_interview\_preparation/**](https://learningpartner.github.io/javaScript_interview_preparation/)

**mini Project Ideas**

**https://miniprojectideas.com/**

**VIDEO: 7\_Angular Batch -29 \_ Recording 18th Apr**

**ngClassStyle**

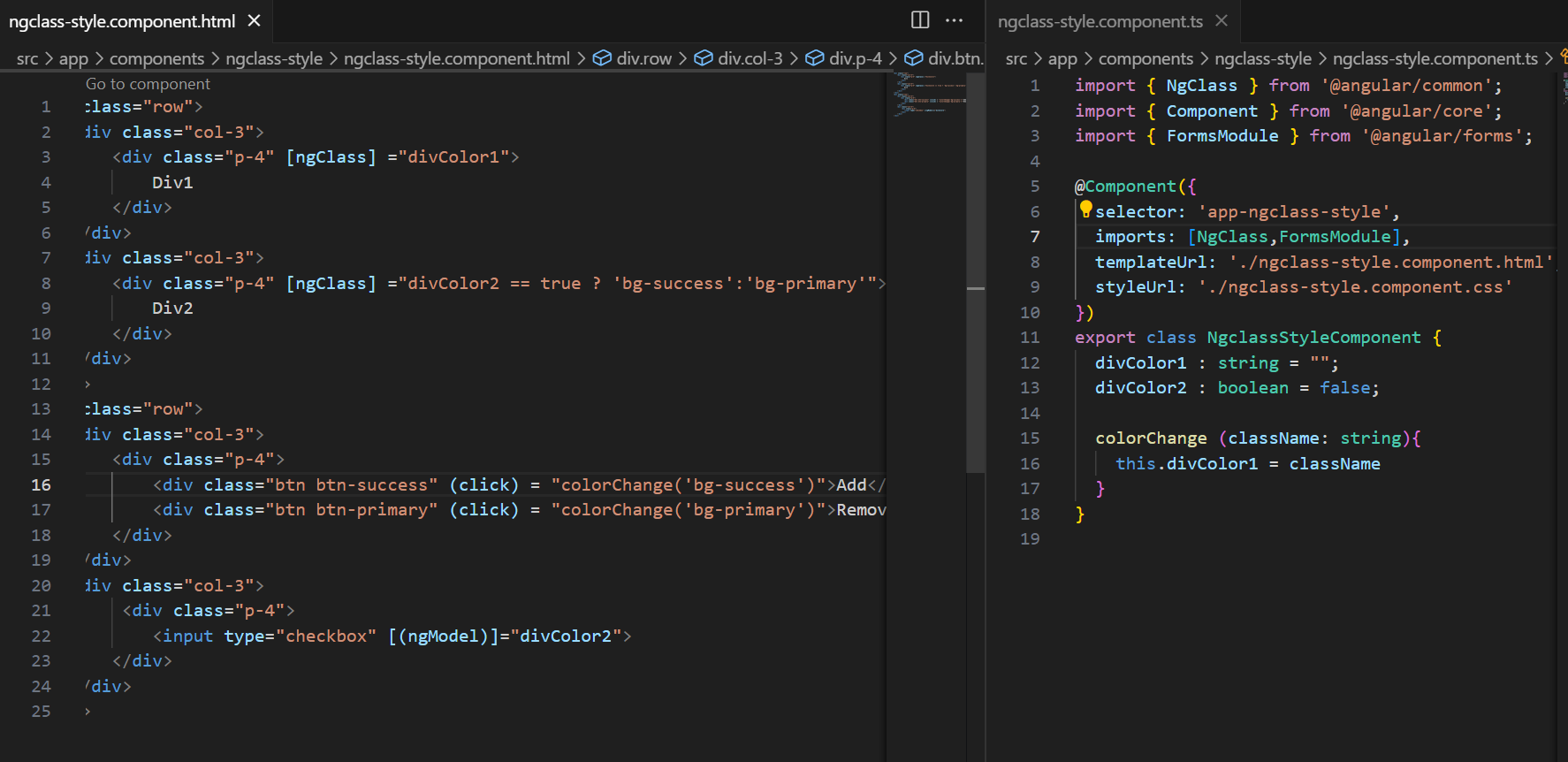
on click of button colour change should takes place.

ngClass is responsible to add dynamic class to the element.

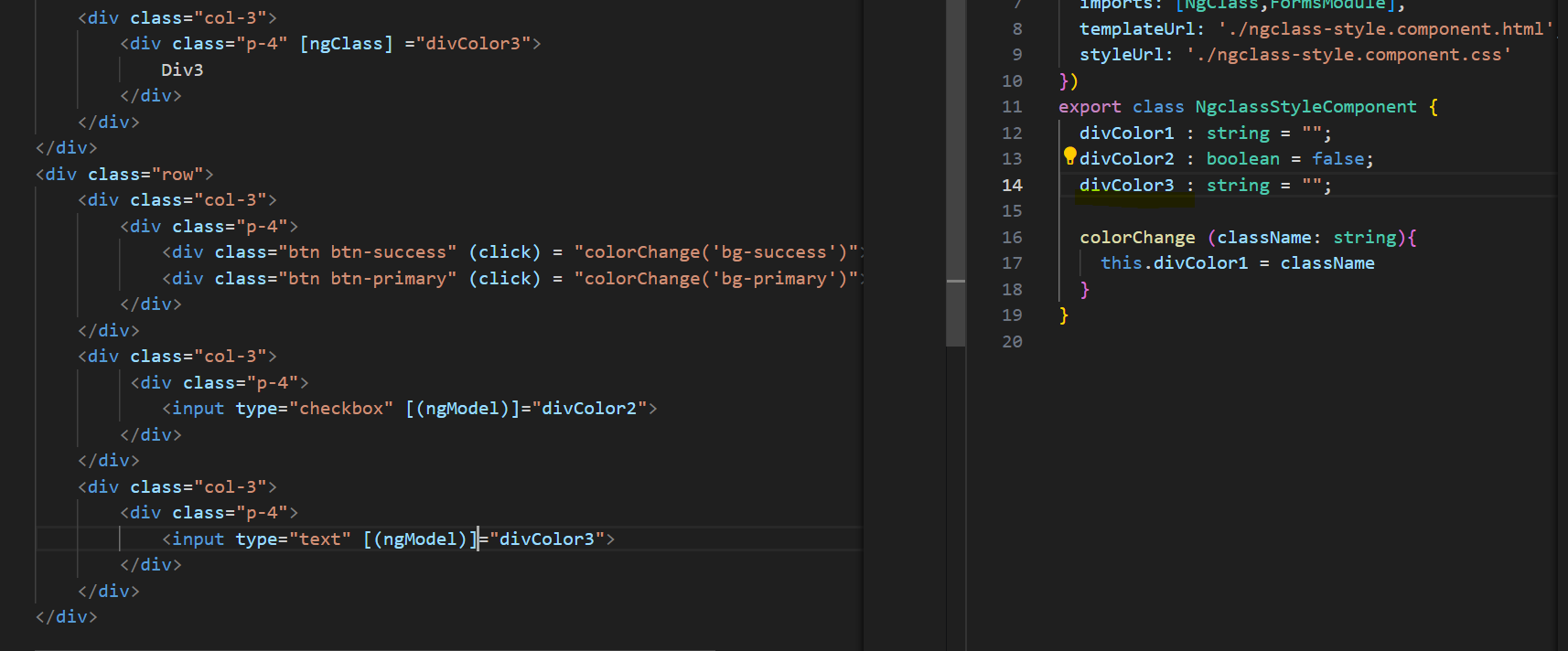
<div class=> **static** but [ngClass]=> **Dynamic**.

If it's a **string** → assign a class name directly.

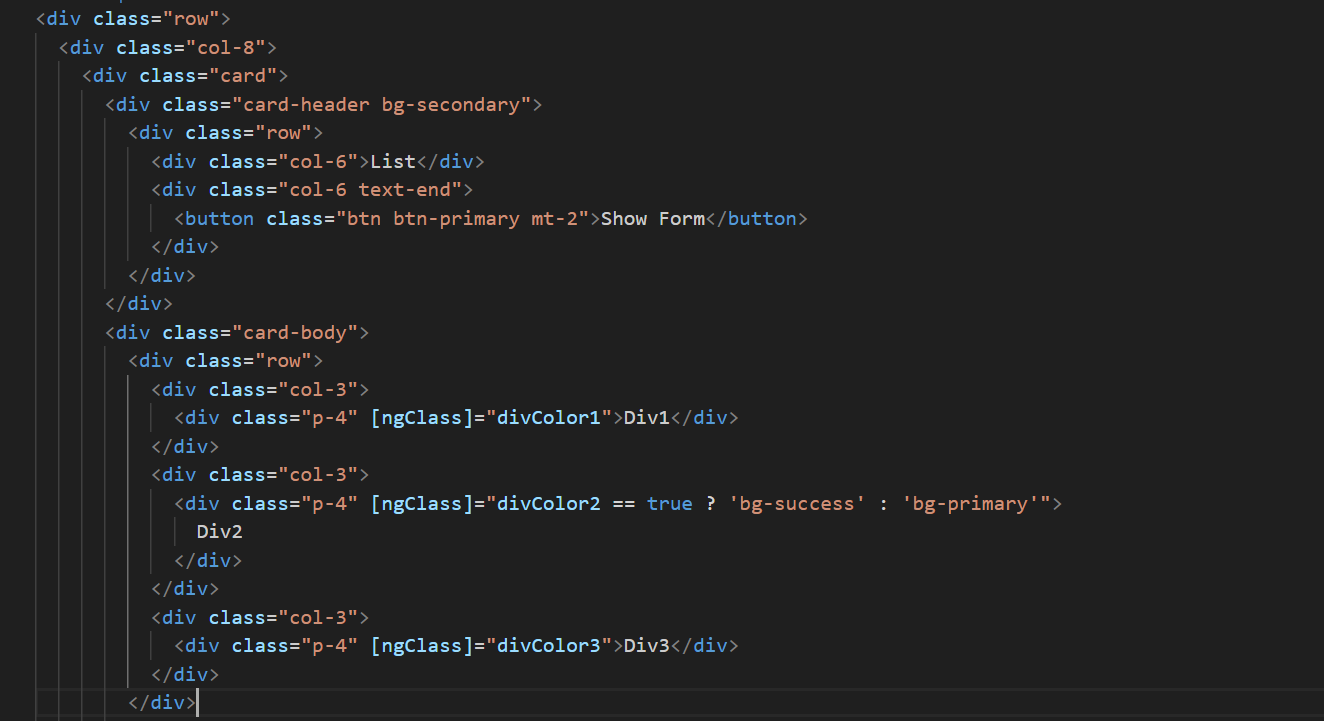
If it's a **boolean** → toggle the class using object syntax.

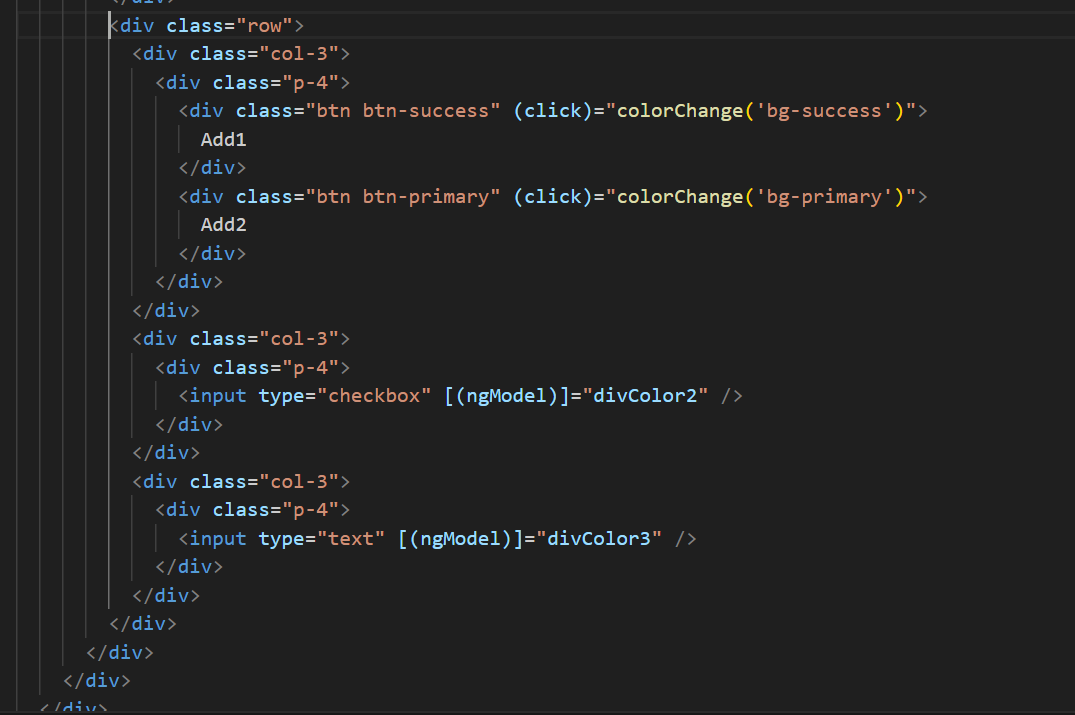
****

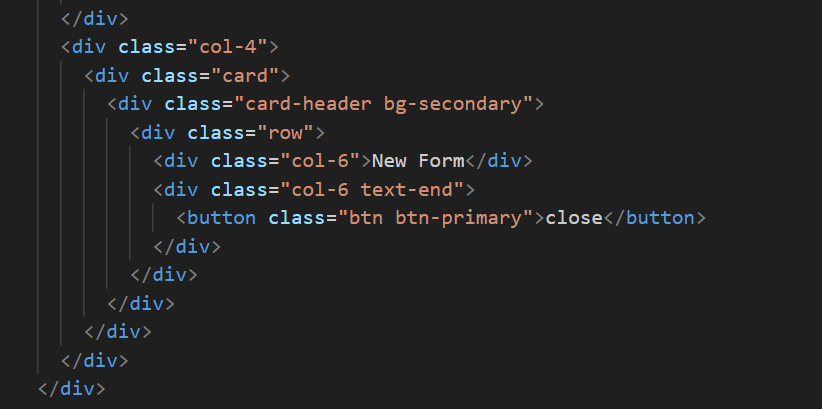
**Dynamically we can add class to the element.**

****

**To get the above output, find the code below.**

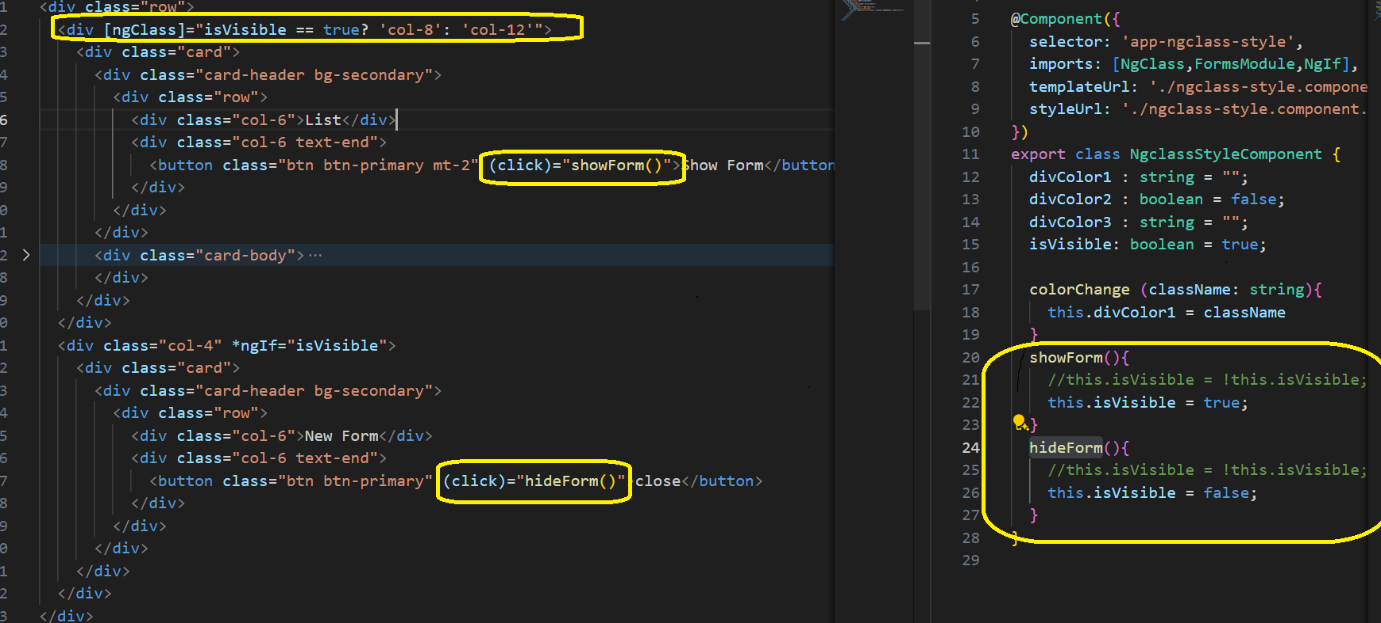
****

****

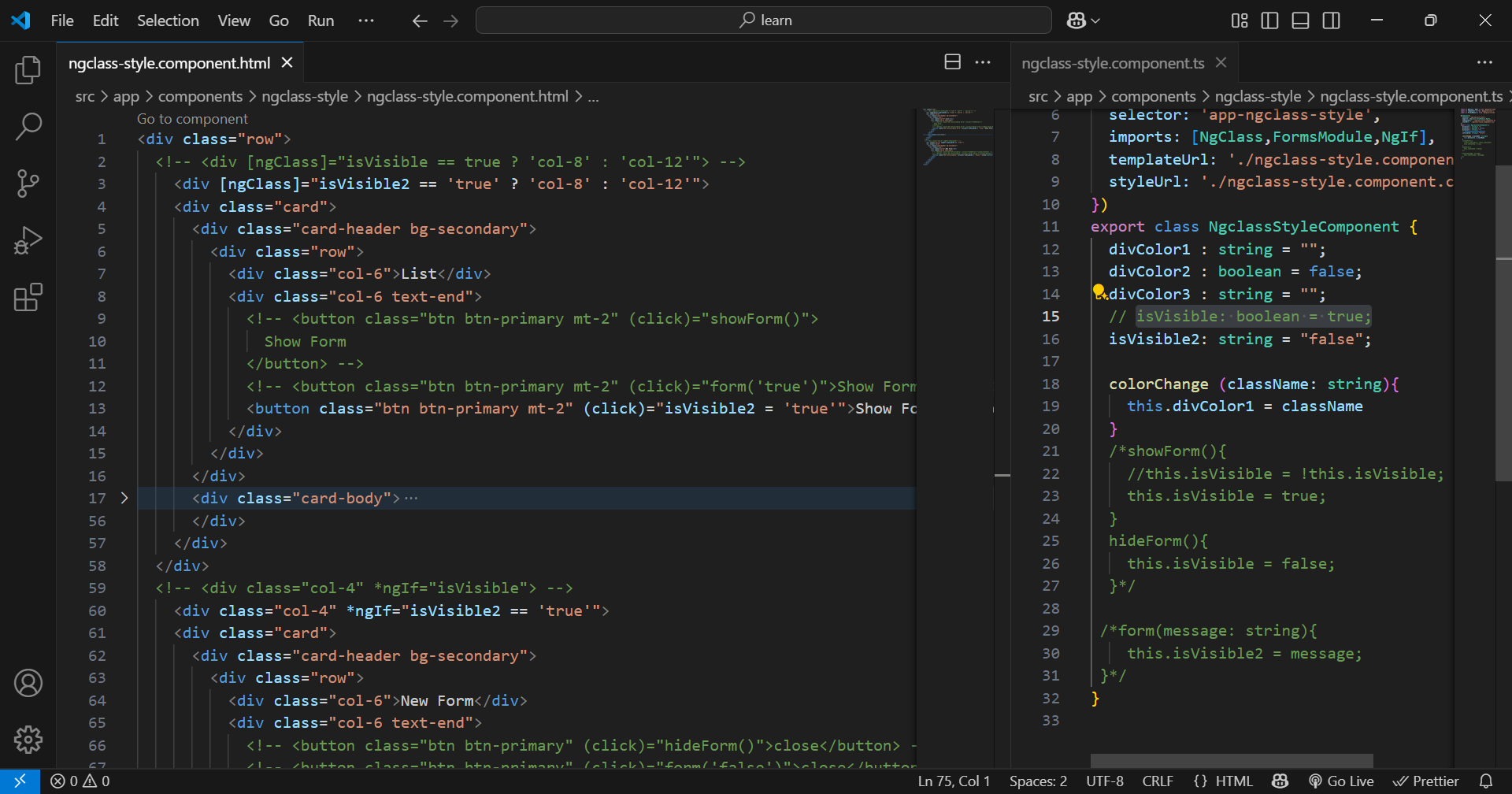
****

**We need to implement a few changes:**

* **When we click the "Show Form" button, the List div should resize, and the New Form should appear alongside it.**
* **When we click the "Close" button in the New Form, the List div should expand to occupy 100% width again.**

****

**We have 3 different ways to achieve it .**

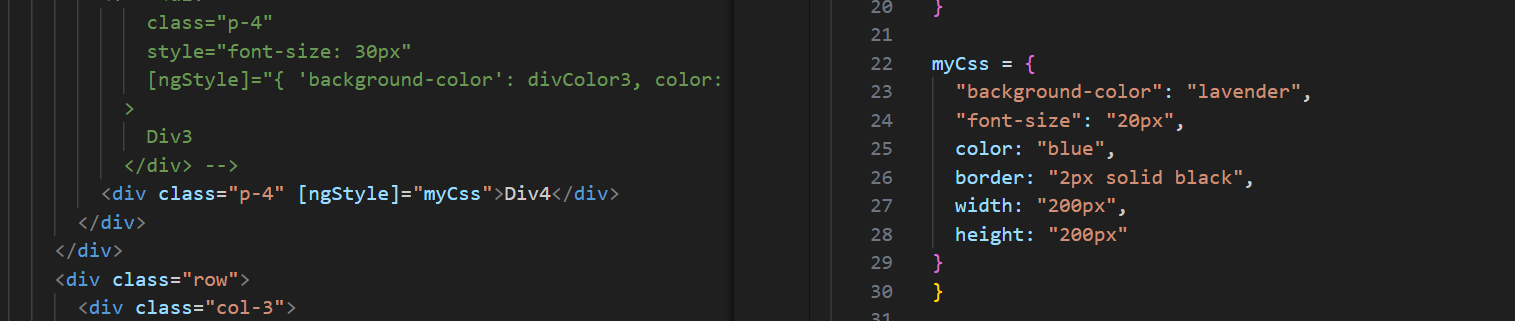


**Dynamically we can add style(background-color) to the element.** 

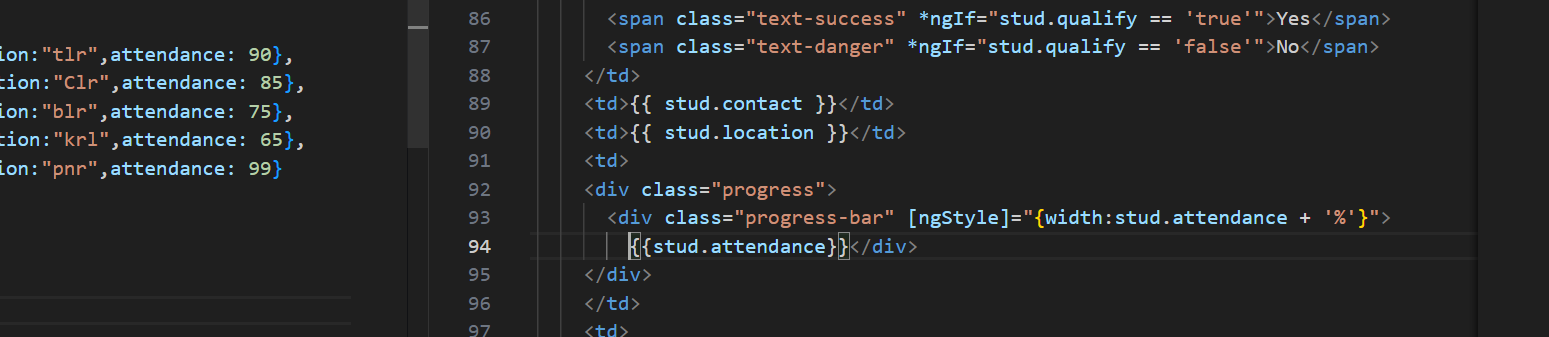
**Hint:** [ngStyle] have more preferences.

****

simple **example** showing how **some APIs return a CSS class** in their response



How to show progress bar in UI: with using bootstrap progress bar we can achieve it.

****

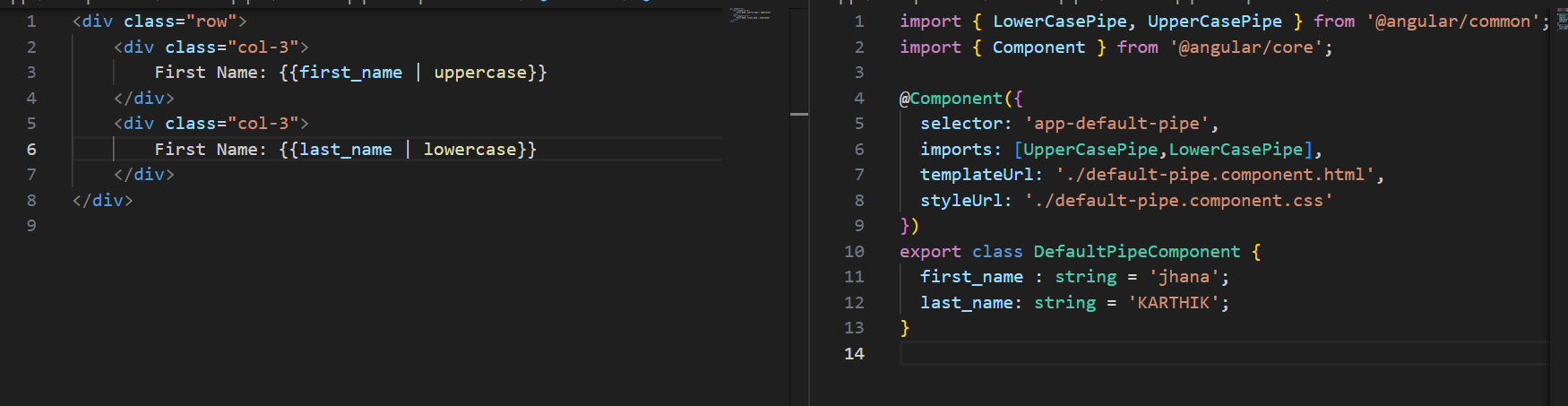
**VIDEO: 8\_GMT20250421-013759\_Recording\_1920x1080**

**PIPE:**

Formatting the data.

**Scenario 1: Declare a name in lowercase, but display it in uppercase and vise versa.**

Using a pipe, we can achieve this. In the .ts file, we import the pipe**(UpperCasepipe, lowerCasePipe OR commonModule)** and in the HTML file, we use the variable name followed by | uppercase.

****

**VIDEO: 9\_GMT20250422-014358\_Recording\_1920x1080**

**VIDEO: 10.GMT20250423-014546\_Recording\_1920x1080**

**VIDEO:11.GMT20250424-014327\_Recording\_1920x1080**

**VIDEO:12.GMT20250425-020413\_Recording\_1920x1080**

**VIDEO:13\_GMT20250428-014059\_Recording\_1920x1080**