Angular app - one or more modules

Module – one or more components and services

Components -HTML+class

Services- business logic

Modules interact and ultimately render the view in the browser

To generate new component:

Ng g c name

**Interpolation:**

Bind data from a class to a template

@Component({

  selector: 'app-test',

  template:`

  <h2>

  welcome to angular {{name}}

</h2>` ,

  styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

  public name ="sai";

{{name}} in above image is called as interpolation

**Dos:**

1.Can do addition:{{2+2}} op 4

2.Concatenation {{“sa”+”I”}} op sai

3.{{name.length}} display length of name JS property

4.{{name.toUpperCase()}} converts lower to upper case and displays op JS property

5.greetUser() {return “welcome”+this.name;

}

{{greetuser}} we can call methods

**Donts:**

1.Assigning expression to variable

{{a=2+2}} gives template parse errors i.e, assignments not possible

2.Unable to access urls In binding i.e,{{}}

**Property binding:**

Attributes and properties are not same

Attributes-html

Properties -DOM(document object model)

Attributes initialize DOM properties and then they are done.attribute values cannot change once they are initialized.

Property values however can change.

<!--by property binding -->

<input [id]="myId" type ="text" value ="sai">

<!-- by interpolation -->

<input id="{{myId}}" type ="text" value ="sai">

` ,

  styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

  public name ="sai";

  public myId="testId";

we can use interpolation also but only string values are acceptable

<input disabled id =”{{myId}}” type=”text” value=”sai”>

From above code, by using interpolation once code is run , the input element will be disabled.bcz automatically the disabled will be true this time.

Next time if you set disabled to false, changes wont reflect on webserver.

<input disabled=”{{false}}” id =”{{myId}}” type=”text” value=”sai”>.

To avoid this , use property binding

<input [disabled]=”false” id =”{{myId}}” type=”text” value=”sai”>

Note:

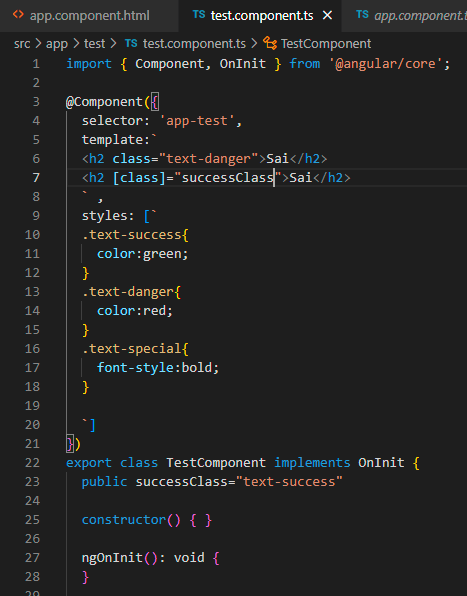
Instead of using [] to property binding, u can use bind-

<input bind-disabled=”false” id =”{{myId}}” type=”text” value=”sai”>

**Class binding:**



[class] is for class binding, find below image



**Note:**

<h2 class=”text-special” [class]=”successClass”>Sai</h2>

Only the text is in Bold format but not in green colour, hence both effects are not applied

We can use one or other , but not both.

**Another syntax for class binding(conditionally apply for a single class)**

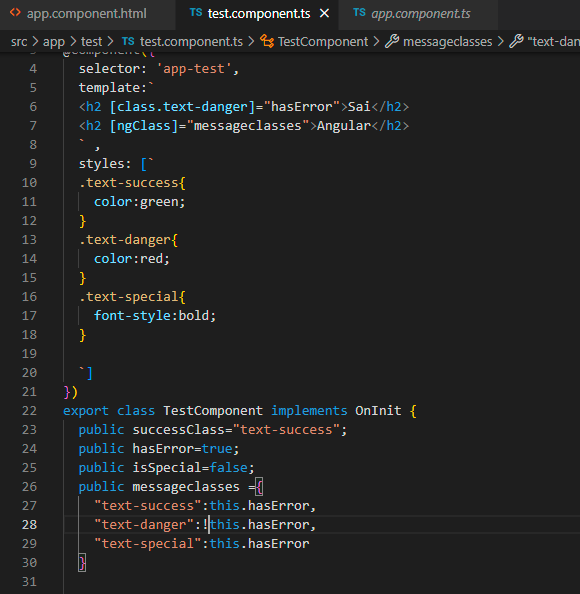
Apply class based on expression that evaluated to be true or false



**Class -binding(conditionally apply for multiple classes):**

Angular provides us with ngClass directive

Directive is nothing but a custom html attribute that angular provides

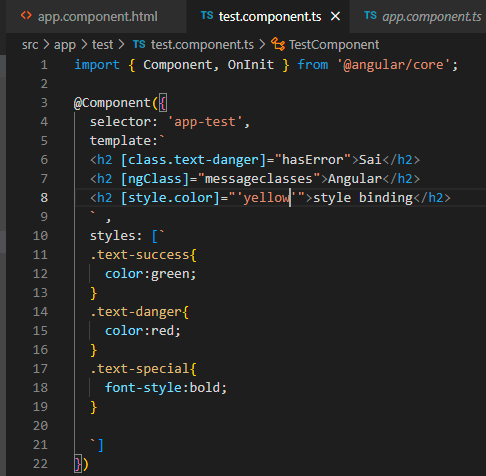


Output: it will be green and bold

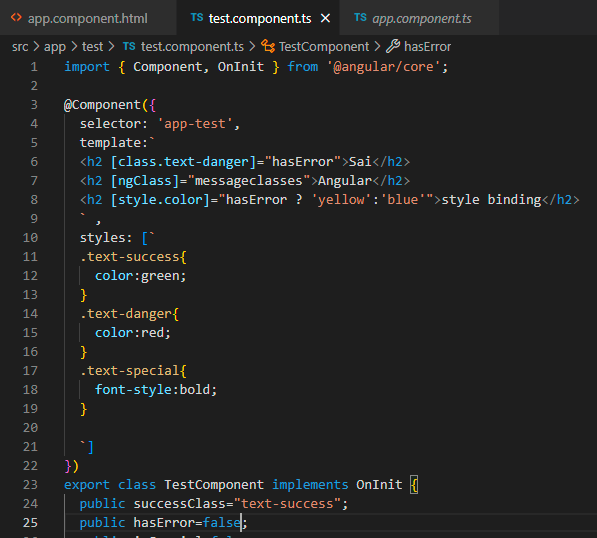
**Style binding:**

In angular, Style binding is useful to apply inline styles to html elements

Its similar to class binding



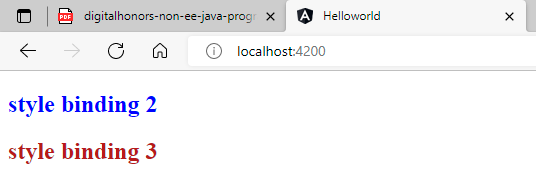
Output:text will be yellow color



Op:blue text

Note: to apply multiple styles, we use “ngStyle” directive provided by angular





**Event binding:**

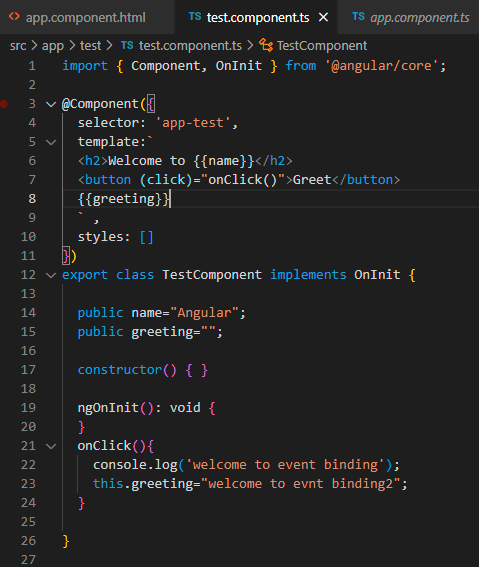
Template

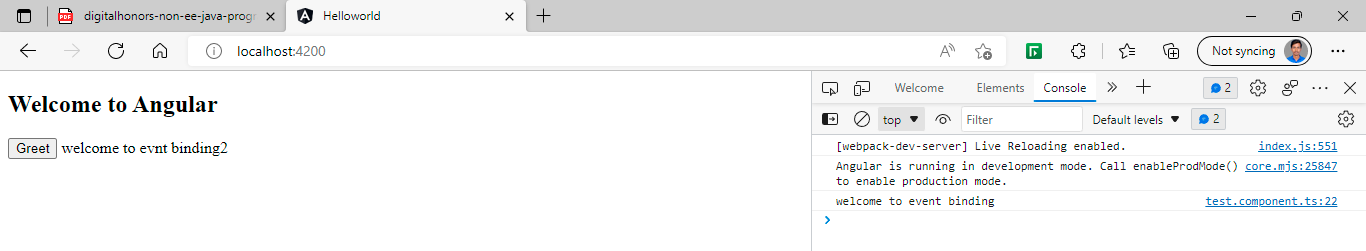
Event bind ()

Data binding

[]

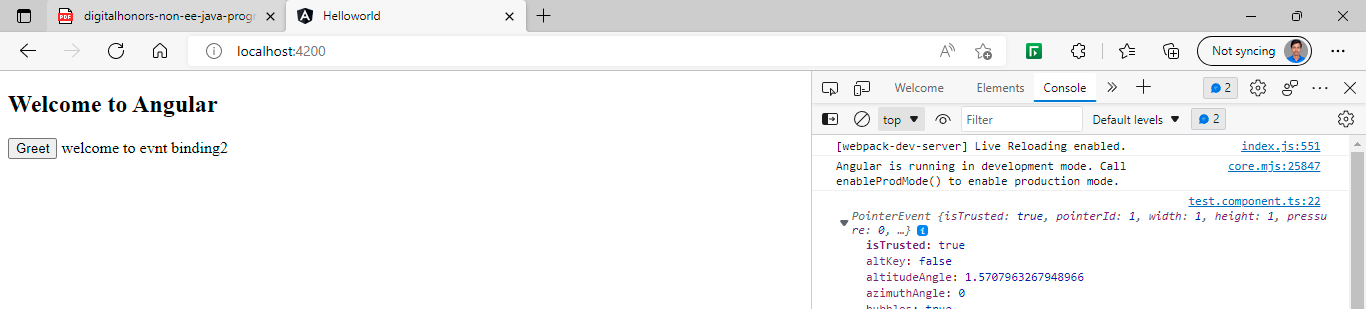
Class

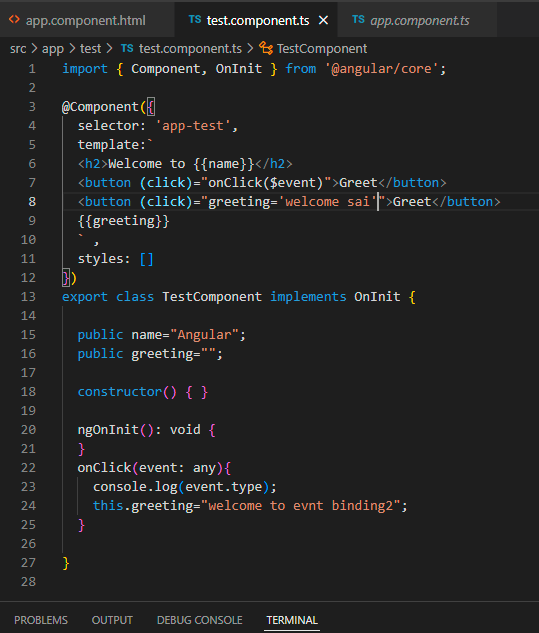


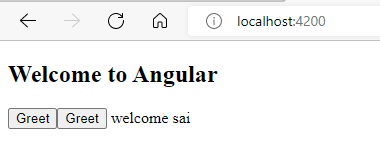


To get information about event itself:







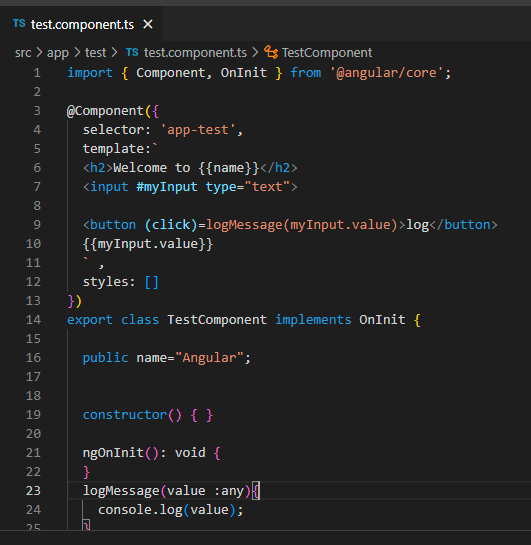


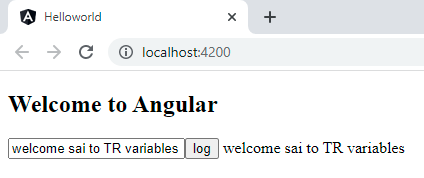
**Template reference variables:**

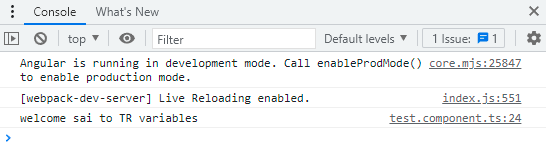
When there is an user interaction , we might want some data to flow from view to class to perform operation

To easily access access DOM elements and their properties, angular provides template reference variables

We need to use # for corresponding input element.







**Two way data binding:**

**ngModel** dir is used to achieve two way data bing

must Import forms module from @angular/forms in app .module.ts

[] – property binding

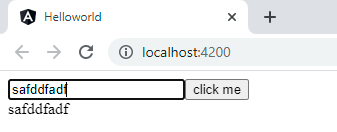
() – event binding

Two way data binding = property binding + event binding

=[]+()

=[()] banana in the box





**ngIf directive:**

**structural directives**

used for adding or removing html elements from the template

commonly used structural directives are

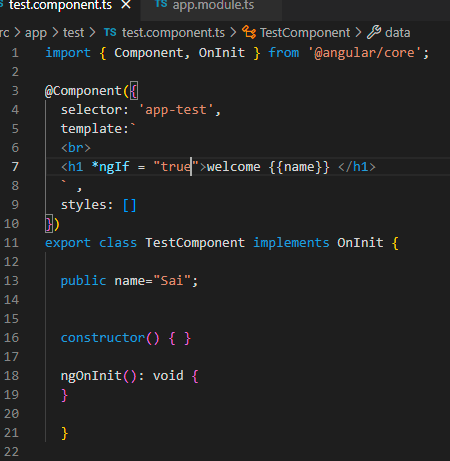
-**ngIf**

**-ngSwitch**

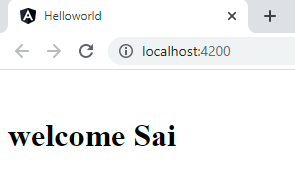
above both used for conditionally render HTML elements

-**ngFor** – used for render list of HTML elements

**ngIf :**



Output name is displayed because the condition is true



**ng-template is used to declare the else block, pfb.**

