What Is Angular Data Binding?

- Data binding in Angular is essentially a way to establish a connection between the application's data and the UI components.
- It involves binding properties of components (such as input fields, labels, etc.) to data sources, like variables or models, so that changes in one automatically reflect in the other.
- Data binding is a mechanism that allows synchronization of data between the model and the view, making it easier to manage and update user interfaces efficiently. There are many applications for Angular databinding.
- It is typically utilized in web applications that have many user interface elements, like forms, calculators, tutorials, games, and movies. The data binding angular procedure is helpful to handle sites with large amounts of data.

Types of Data Binding

- Angular allows both One-way and Two-way Data Binding. We will study both of them in detail in the below section.

  1. One-Way Binding

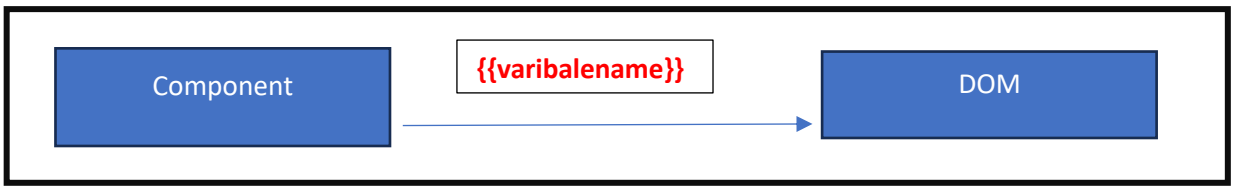
  In one-way binding, data flows in a single direction, either from the component to the view (interpolation or property binding) or from the view to the component (event binding).

  a. Interpolation

  This involves displaying component data in the view by enclosing the property or expression in double curly braces, like {{ data }}. Angular automatically updates the view whenever the underlying data changes.

  Syntax

  "{{variable_name}}"




```
<h3>Binding Types</h3>
<p>Interpolation</p>
<br>
<h5>
        Subtraction of 8 from 15 with
        Interpolation is {{15-8}}
</h5>
<h5>
        Subtraction of 8 from 15 with
        Interpolation is 15-8
</h5>
<h2>{{val}}</h2>
```

```
import { Component } from '@angular/core';
@Component({
        selector: 'my-app',
        templateUrl: './app.component.html',
        styleUrls: ['./app.component.css'],
})
export class AppComponent {
        val: string = 'Welcome to TQ';
}
```

2. Property Binding

This is achieved by using square brackets to bind a property of an HTML element to a component property. For instance, [property]="data" binds the value of the component's "data" property to the HTML element's property.

Syntax

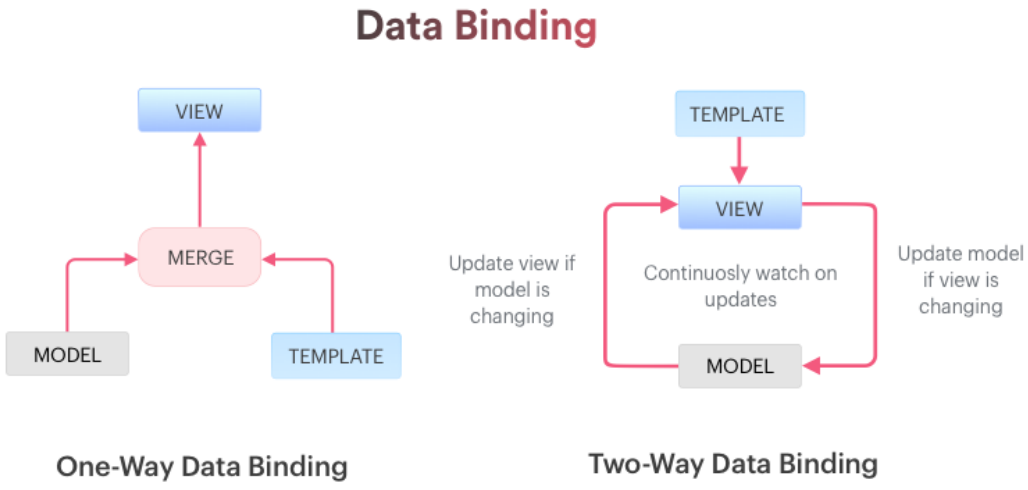
[class]="variable_name"

3. Event Binding

This allows the view to communicate changes back to the component when an event occurs, such as a button click or input change. Event binding is denoted by parentheses, like (event)="handler()".

2. Two-Way Binding

- Two-way binding is a combination of both one-way binding for property binding and event binding. It simplifies the synchronization between the component and the view by using the ngModel directive, allowing changes in the view to update the component and vice versa.
- Here, the immediate changes to the view & component will be reflected automatically, i.e. when the changes are made to the component or model then the view will render the changes simultaneously.
- Similarly, when the data is altered or modified in the view then the model or component will be updated accordingly.



Data Binding In Angular

Attribute Binding

- In our **Angular Interpolation** and **Property Binding** articles**,** we discussed that they both (Interpolation and Property Binding) are dealing with the DOM Properties but not with the HTML attributes.
- But there are some HTML elements (such as colspan, area, etc) that do not have the DOM Properties.
- Now the question is how to deal with such elements that do not have DOM Properties as we can't use Interpolation and Attribute Binding. The Answer is Angular Attribute Binding.
- With Attribute Binding in Angular, you can set the value of an HTML Element Attribute directly. So, the Attribute Binding is used to bind the attribute of an element with the properties of a component dynamically.

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

    ColumnSpan: number = 2;

    pageHeader: string = 'Student Details';
    FirstName: string = 'Anurag';
    LastName: string = 'Mohanty';
    Branch: string = 'CSE';
    Mobile: number = 9876543210;
    Gender: string = 'Male';
    Age: number = 22;
}
```

```
<thead>
    <tr>
        <th [attr.colspan]="ColumnSpan">
            {{pageHeader}}
        </th>
    </tr>
</thead>
```

Class Binding

- The Angular Class Binding is basically used to add or remove classes to and from the HTML elements.
- It is also possible in Angular to add CSS Classes conditionally to an element, which will create the dynamically styled elements and this is possible because of Angular Class Binding.
- The class binding syntax is also like property binding. In property binding, we only specify the element between brackets.
- But in the case of class binding, it starts with the prefix class, followed by a dot (.), and the name of the class. You then bind the class value with CSS class name like **[class.class-name]**.
- **<h1 [class.class-name]="Class Value"></h1>**

Style Binding in Angular

- Style binding is used to set a style of a view element. We can set inline styles with style binding.
- Like with class and attribute binding, style binding syntax is like property binding.
- In property binding, we only specify the element between brackets. But in case of style binding, it starts with the prefix class, followed by a dot (.) and the name of the style.
- You then bind the style value with CSS style name like the **[style.style-name]**.

```
<div>
<h1 [style.color]="blue">This is a Blue Heading</h1>
</div>
```

Templates Variable in Angular

- Template variables in Angular allow us to use the data from one part of the template to another part of the same template.
- There are various use cases where these template variables come in handy. For example, responding to the user input or responding to the form submission.
- In the template, we use hash symbol (#) to declare the template variable. In the below syntax we've declared a template variable called "templateVarName" for the input element. Then the button element is referring to this variable and passes its value to the handleSubmit() method.

```
<input #templateVarName />
<button type="button" (click)="handleSubmit(templateVarName.value)">Submit</button>
```