**DATA BINDING IN ANGULAR**

## **What Is Angular Data Binding?**

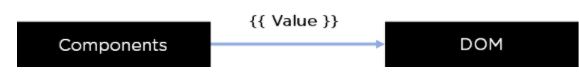
Data binding allows Internet users to manipulate web page elements with the help of a web browser. It includes dynamic HTML and does not require complex programming. Data binding is used in web applications that contain interactive components such as forms, calculators, tutorials, and games. The incremental display of a webpage makes data binding convenient when pages contain an extensive amount of data.

Angular uses the concept of two-way binding. Any UI element-related change is reflected in the corresponding and specific model state. Conversely, any model state changes reflect in the UI state. This ensures that the framework is able to connect the DOM to the Model data with the help of the controller.

## **Types of Data Binding**

### **Interpolation Binding**

Interpolation is a procedure that allows the user to bind a value to the user interface element. Interpolation binds the data one-way, which means that data moves in one direction from the components to HTML elements.



Graphical user interface, application

Description automatically generated

### **Property Binding**

Property binding is a one-way data binding mechanism that allows you to set the properties for HTML elements. It involves updating a property value in the [component](https://www.simplilearn.com/tutorials/angular-tutorial/angular-components) and binding the value to an HTML element in the same view. We use property binding for toggle functionality and sharing data between components. It uses the "[]" syntax for data binding.

A screenshot of a computer

Description automatically generated

### **Event Binding**

Event binding type is when information flows from the view to the component when an event is triggered. The event could be a mouse click or keypress. The view sends the data to update the component. Unsurprisingly, it is the exact opposite of property binding, where the data goes from the component to the view.

view-event

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## **Two-way Data Binding**

As the name suggests, two-way binding is a mechanism where data flows from the component to the view and back. This binding ensures that the component and view are always in sync. Any changes made on either end are immediately reflected on both. The general syntax to denote two-way data binding is a combination of Square brackets and parentheses "[()]".

2-way

To illustrate two-way data binding, we’ve created a property with an empty string and an input box for the user to type. Whatever the user provides is displayed on the screen with the help of the property.

In the .ts file,

public random = ""

In the .html file, we’ve created an input field

<input [(ngModel)]="random" type="text"> <br>

  {{random}}

We have used the ngModel directive and initialized it to random. We’ve then interpolated the property random.

When we use a ngModel directive inside the form input field, we will see this type of error in our terminal.

A screenshot of a computer

Description automatically generated

<input type="name" [(ngModel)]="name"/>

Error

ERROR in src/app/app.component.html:3:41 - error NG8002:

Can't bind to 'ngmodel' since it isn't a known property of 'input'.

<input type="name" [(ngmodel)]="name" />

~~~~~~~~~~~~~~~~~~~

src/app/app.component.ts:6:16

6 templateUrl: './app.component.html',

~~~~~~~~~~~~~~~~~~~~~~

Error occurs in the template of component AppComponent.

This occurs due to a FormsModule is not injected into the angular application.

**Fixing the error**

To fix this error, open your app.module.ts file and import the FormsModule from @angular/forms package and add it to the imports array.

app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

declarations: [

AppComponent,

],

imports: [

BrowserModule,

FormsModule ],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

Graphical user interface, application

Description automatically generated