Data Binding in Angular

Data binding is the core concept of Angular 8 and used to define the communication between a component and the DOM. It is a technique to link your data to your view layer. In simple words, you can say that data binding is a communication between your typescript code of your component and your template which user sees. It makes easy to define interactive applications without worrying about pushing and pulling data.

Data binding can be either one-way data binding or two-way data binding.

One-way databinding

One way databinding is a simple one way communication where HTML template is changed when we make changes in TypeScript code.

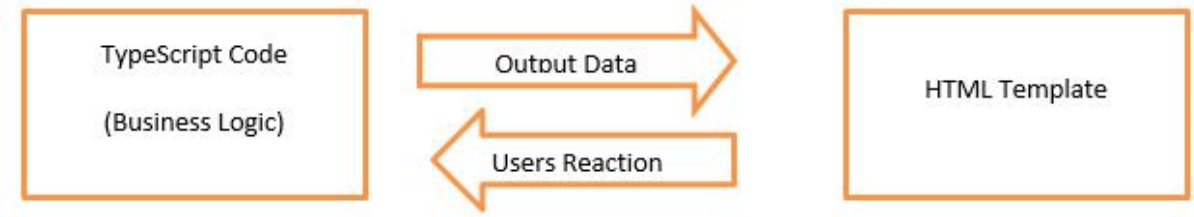
Or

In one-way databinding, the value of the Model is used in the View (HTML page) but you can't update Model from the View. Angular Interpolation / String Interpolation, Property Binding, and Event Binding are the example of one-way databinding.

Two-way databinding

In two-way databinding, automatic synchronization of data happens between the Model and the View. Here, change is reflected in both components. Whenever you make changes in the Model, it will be reflected in the View and when you make changes in View, it will be reflected in Model.

This happens immediately and automatically, ensures that the HTML template and the TypeScript code are updated at all times.



Angular provides four types of data binding and they are different on the way of data flowing.

- **String Interpolation**
- **Property Binding**
- **Event Binding**
- **Two-way binding**

String interpolation

String Interpolation is a **one-way databinding technique** which is used to output the data from a TypeScript code to HTML template (view). It uses the template expression in **double curly braces** to display the data from the component to the view.

For example:

{{ data }}

String interpolation adds the value of a property from the component:

We have already created an Angular project using Angular CLI.

Open **app.component.ts** file and use the following code within the file:

```
//String Interpolation is a one-way databinding
  title = 'Data binding example using String Interpolation';
```

Now, open **app.component.html** and use the following code to see string interpolation.

```
<h2>
  {{ title }}
</h2>
```

String Interpolation can be used to resolve some other expressions too. Let's see an example.

Update the **app.component.ts** file with the following code:

```
numberA: number = 10;
numberB: number = 20;
```

app.component.html:

```
<h2>Calculation is : {{ numberA + numberB }}</h2>
```

Property Binding

Property Binding is also 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.

For example:

****

Property binding is preferred over string interpolation because it has shorter and cleaner code String interpolation should be used when you want to simply display some dynamic data from a component on the view between headings like h1, h2, p etc.

Note: String Interpolation and Property binding both are one-way binding. Means, if field value in the component changes, Angular will automatically update the DOM. But any changes in the DOM will not be reflected back in the component.

Open **app.componnt.ts** file and add the following code:

```
    imgUrl="assets/images/angular.jpg" ;
```

Now, open **app.component.html** and use the following code for property binding:

```
<img [src]="imgUrl" height="250px" /> <!-- Property Binding -->
```

Event Binding

In Angular 13, event binding is used to handle the events raised from the DOM like button click, mouse move etc. When the DOM event happens (eg. click, change, keyup), it calls the specified method in the component.

Now, open the **app.component.ts** file and use the following code:

```
  onSave($event:any){
      console.log("Save button is clicked!", $event);
    }
```

app.component.html:

```
<h2> Event Binding Example</h2>
<button (click)="onSave($event)">Save</button> <!--Event Binding-->
```

Two-way Data Binding

We have seen that in one-way data binding any change in the template (view) were not be reflected in the component TypeScript code. To resolve this problem, Angular provides two-way data binding. The two-way binding has a feature to update data from component to view and vice-versa.

In two way data binding, property binding and event binding are combined together.

Note: For two way data binding, we have to enable the ngModel directive. It depends upon FormsModule in angular/forms package, so we have to add FormsModule in imports[] array in the AppModule.

Open your project's **app.module.ts** file and use the following code:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import {FormsModule} from '@angular/forms';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts file:

```
fullName: string = "Hello Angular";
```

app.component.html file:

```
<h2>Two-way Binding Example</h2>
   <input [(ngModel)]="fullName" /> <br/><br/>
<p> {{fullName}} </p>
```

(Directives)

- Directives enhances the view capabilities.
- We have two types of directives
    - Pre-defined directives
    - Custom directives
- The directives are given by angular framework is called predefined directives.
- The directives are doveloped by us based on application requirement called as custom directives

=>Pre-defined directives

1. ngFor
2. ngif
3. (click)
4. (dbclick)
5. [(ngmodel)]
6. (ngsubmit)
7. [ngclass]
8. [ngstyle]
9. [ngswitch]

- Directives are categorized into three types
  - Structural type directives
  - Event type directives
  - Attribute type directives

- Directives are categorized into three types
  - Structural type directives
  - Event type directives
  - Attribute type directives

- Structural type directives have manipulate into dom
- Structural type directives starts with "*"
- Based on the requirement we are adding or removing dom elements from browser memory.

- In order to handle events raised by dom ,we are using event type directives.
- Event type directives are serounder with “()”
- Attribute type directives serounder with ”[]”

1) *ngFor

- this directive used to iterate the Array Elements.

Syntax.

*ngFor= "let variable of array;constant1,constant2,...."

constants

---

1) index

- it is used to get the indexes for each iteration.

2) **first**

- it is used to recognise the first element in array.

3) last

- it is used to recognise the last element in array.

4) even

- it will recognise even positions in array.

5) odd

- it will recognise odd positions in array.

2) *ngIf

- this directive helps to write the conditions.

2) *ngIf

- this directive helps to write the conditions.

Directory Structure:

```
*****************************************

firstApp

        src

       app

         first.component.ts

         first.component.html

         app.module.ts

       index.html

*****************************************
```

Adding Bootstrap

Angular.json

```
"styles": [
            {
              "input": "./node_modules/bootstrap/dist/css/bootstrap.min.css"
            },
            "src/styles.css"
          ],
```

PS C:\Shekhar\Angular2> npm install --save bootstrap@5.1