Data Binding in Angular

16 Comments / 7 minutes of reading / March 9, 2023

← Angular Components | Angular Tutorial | Interpolation →

In this tutorial, we are going to look at the How Data Binding works in Angular with examples. Angular Components are useless if they do not show any dynamic data. They also need to respond to user interactions and react to events. The data binding kees both component & view in sync with each other. We use techniques like Interpolation, Property Binding, Event Binding & Two Way Binding to bind data. We also learn how to use the ngModel directive to achieve the two-way binding in Angular Forms.

Table of Contents

What is Angular Data Binding

Data binding is a technique, where the data stays in sync between the component and the view. Whenever the user updates the data in the view, Angular updates the component. When the component gets new data, the Angular updates the view.

There are many uses of data binding. You can show models to the user, dynamically Change element style, respond to user events, etc

Data Binding in Angular

The data binding in Angular can be broadly classified into two groups. *One way binding* or *two-way binding*

One way binding

In one way binding data flows from one direction. Either from view to component or from component to view.

From Component to View

To bind data from component to view, we make use of Interpolation & Property Binding.

Interpolation

Interpolation allows us to include expressions as part of any string literal, which we use in our HTML. The angular evaluates the expressions into a string and replaces it in the original string and updates the view. You can use interpolation wherever you use a string literal in the view

The Angular uses the {{ }} (double curly braces) in the template to denote the interpolation. The syntax is as shown below

{{ templateExpression }}

The content inside the double braces is called **Template Expression**

The Angular first evaluates the Template Expression and converts it into a string. Then it replaces Template expression with the result in the original string in the HTML. Whenever the template expression changes, the Angular updates the original string again

Example

```
1
2  Welcome,  {{firstName}} {{lastName}}
3
```

```
1
2  import { Component } from '@angular/core';
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css']
8  })
9  export class AppComponent {
10   firstName= 'Sachin';
11   lastName=”Tendulkar”
12 }
13
14
```

Run the app and you will see `Welcome, Sachin Tendulkar` in the output. The Angular replaces both `{{firstName}}` & `{{lastName}}` with the values of `firstName` & `lastName` variable from the component.

Also, whenever the values of `firstName` & `lastName` change, Angular updates the view. But not the other way around.

Read More

Interpolation in Angular

Property binding

The Property binding allows us to bind HTML element property to a property in the component. Whenever the value of the component changes, the Angular updates the element property in the View. You can set the properties such as class, href, src, textContent, etc using property binding. You can also use it to set the properties of custom components or directives (properties decorated with @Input).

The Property Binding uses the following Syntax

```
[binding-target]="binding-source"
```

The `binding-target` (or target property) is enclosed in a square bracket `[]`. It should match the name of the property of the enclosing element.

`Binding-source` is enclosed in quotes and we assign it to the `binding-target`. The Binding source must be a template expression. It can be property in the component, method in component, a template reference variable or an expression containing all of them.

Whenever the value of Binding-source changes, the view is updated by the Angular.

Example

app.component.html

```
<h1 [innerText]="title"></h1>
<h2>Example 1</h2>
<button [disabled]="isDisabled">I am disabled</button>
```

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title="Angular Property Binding Example"

  //Example 1
  isDisabled= true;

}
```

The title property of the component class is bound to the innerText property of the h1 tag. Disabled Property of the button is bound to the isDisabled Property of the component

Whenever we modify the title or isDisabled in the component, the Angular automatically updates the view.

The property binding has special syntaxes for setting the class & styles. Also, both interpolation & property binding does not set the attributes of the HTML elements.

Hence we have an attribute binding to such situations

Read More

Angular Property Binding

Class Binding

You can set the class in the following ways. Click on the links to find out more

- ClassName Property binding
- Set the Class attribute with class binding
- ngClass directive

Style Binding

You can set the class in the following ways. Click on the links to find out more

- Style Binding
- ngStyle directive

Attribute binding

Sometimes there is no HTML element property to bind to. The examples are aria (accessibility) Attributes & SVG. In such cases, you can make use of attribute binding

The attribute syntax starts with `attr` followed by a `dot` and then the name of the attribute as shown below

```
1
2 <button [attr.aria-label]="closeLabel" (onclick)="closeMe()">X</button>
3
```

Read More

1. ngClass Directive
2. Style Binding
3. ngStyle directive

From View to Component

Event Binding

Event binding allows us to bind events such as keystrokes, clicks, hover, touch, etc to a method in component. It is one way from view to component. By tracking the user events in the view and responding to it, we can keep our component in sync with the view. For Example, when the user changes input in a text box, we can update the model in the component, run some validations, etc. When the user submits the button, we can then save the model to the backend server.

Angular uses the following syntax for event binding

```
<target-event)="TemplateStatement"
```

Angular event binding syntax consists of a target event name within parentheses on the left of an equal sign, and a quoted template statement on the right.

For Example,

```
1
2  <button (click)="onSave()">Save</button>
3
```

The above example binds the click event of a button to a onSave() method in the component class. Whenever the user clicks on the button, the Angular invokes the `onSave()` method.

Read More

Event Binding in Angular

Two Way binding

Two-way binding means that changes made to our model in the component are propagated to the view and that any changes made in the view are immediately updated in the underlying component

Two-way binding is useful in data entry forms. Whenever a user makes changes to a form field, we would like to update our model. Similarly, when we update the model with new data, we would like to update the view as well

The two-way binding uses the special syntax known as a banana in a box `[()]`

```
<someElement [(someProperty)]="value"></someElement> .
```

The above syntax sets up both property binding & event binding. But to make use of it, the property must have the change event with the name `<propertyName>Change`

But, angular has a special directive ngModel, which sets up the two-way binding

ngModel

The Angular uses the `ngModel` directive to achieve the two-way binding on HTML Form elements. It binds to a form element like `input`, `select`, `selectarea`. etc.

The `ngModel` directive is not part of the Angular Core library. It is part of the `@angular/forms`. You need to import the FormsModule package into your Angular module.

```
1
2 import { FormsModule } from '@angular/forms';
3
```

Then you can use it using the two-way binding syntax as shown below

```
1
2 <input type="text" name="value" [(ngModel)]="value">
3
```

When you bind to a `ngModel` directive, behind the scene it sets up property binding & event binding. It binds to the `value` property of the element using property binding. It then uses the `ngModelChange` event to sets up the event binding to listen to the changes to the value.

Read More

Two way binding in Angular

Summary

The Data binding in Angular consists of interpolation & property binding which is one way from component to view. Interpolation allows us to embed an expression in a