One-Way And Two-Way Data Binding With Examples In Angular

In Angular 2, Data Binding is mainly classified in two ways, namely One-Way data binding (i.e. unidirectional binding) and two-way data binding (i.e. bi-directional binding).

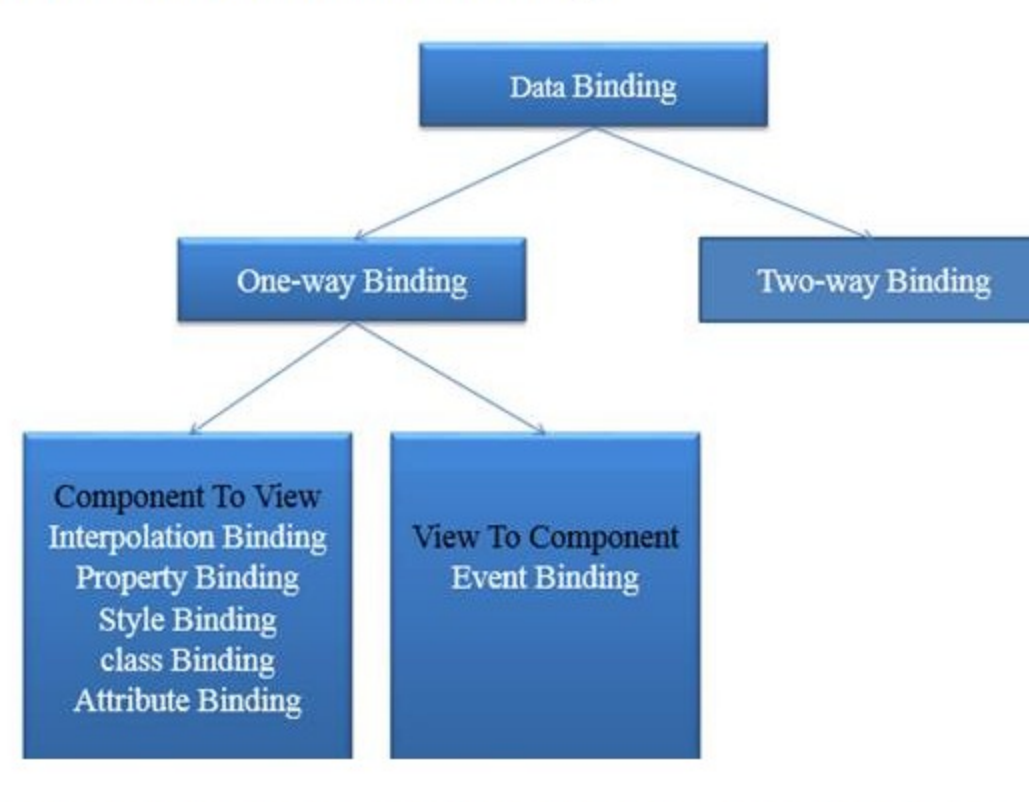
Karthik Elumalai Nov 26 2018

TRENDING UP
01 Transform Classic SharePoint Pages To Modern Look And Feel
02 Create And Build A Skype Bot Using Microsoft Bot Framework
03 SharePoint Framework - Integrating Adaptive Cards With SPFx
04 Visual Studio 2019 New Features
05 Overview Of SharePoint Modernization Scanner
06 SharePoint Framework - Handle Web Part Property Changes
07 Working With AI-Assisted IntelliCode In VS 2017 And VS Code
08 Getting Ready For The SharePoint Modernization Journey
09 SharePoint Framework - Test SPFx WebParts on Modern Pages
10 What Skills Do I Need To Become A Blockchain Developer
View All

Brief of Data Binding

- In Angular 2, Data Binding is mainly classified in two ways - one-way binding (i.e. unidirectional binding) and two-way binding (i.e. bi-directional binding).
- In simple words, if you compare this with MVC applications, it is similar to the process of how we synchronize the data between the View to Model and Model to View.

Classification of Data Binding



In the below sections, we will see both the concepts one by one with examples.

One-way Data Binding: [Component to View]

It will bind the data from Component to View using the following different ways.

Different types of one-way data binding

- Interpolation Binding
- Property Binding
- Attribute Binding
- Class Binding
- Style Binding

Interpolation Binding

- With interpolation, we place the component property name in the View template, enclosed in double curly braces: {{property Name}}.
- In simple words, interpolation is nothing but how we use this binding expression {{}} in our project. We will see that with an example.

Simple Example

File Name: Index.html [start up page is used for all components]

```
<!DOCTYPE html>
<html> e
  <head>
    <title>First Angular Program</title>
    <base href="/src/">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="styles.css">

    <!-- Polyfill(s) for older browsers -->
    <script src="/node_modules/core-js/client/shim.min.js"></script>

    <script src="/node_modules/zone.js/dist/zone.js"></script>
    <script src="/node_modules/systemjs/dist/system.src.js"></script>

    <script src="systemjs.config.js"></script>
    <script>
      System.import('main.js').catch(function(err){ console.error(err); });
    </script>
  </head>

  <body>

      <!--Here is the selector mapped-->
      <my-app>Loading AppComponent content here ...</my-app>

  </body>
</html>
```

File Name: app.component.ts

```
import { Component } from "@angular/core";
@Component({
    selector: 'my-App',
    template: `
                <div>
                <strong>{{firstname}}</strong>
                 <strong>{{lastname}}</strong>
                </div>
})
export class AppComponent {
    firstname: string = "Sachin";
    lastname:string = "Tendulkar"
}
```

Output

Sachin

Tendulkar

Property Binding

- It is used to bind values of component/model properties to the HTML element.
- Depending on the values, it will change the existing behavior of the HTML element.

Syntax

[property] ='expression'

In property binding, there is source and target. For this example, we can define it as [innerHTML] = 'firstname'. Here, innerHTML is a target that is a property of span tag and source is a component property i.e. firstname.

Example

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

@Component({

    selector: 'my-App',
    template: `
                <div>
                <span [innerHTML]='firstname'></span>
                </div>
})
export class AppComponent {
    firstname: string = "Sachin";
}
```

Output

Sachin

Attribute Binding

- With attribute binding, we can set the value of an attribute directly. The thing to note is that you must use the attribute binding only when there is no element property there to bind.
- For example, here are the elements which don't have the property => ARIA, SVG, and table span attributes.

Example

```
<table border="1">
    <thead>
        <tr>
            <th [attr.colspan]="2">Student Details</th>

        </tr>
    </thead>
    <tbody>
        <tr>
            <td>First Name</td>
            <td>{{firstName}}</td>
        </tr>
        <tr>
            <td>Last Name</td>
            <td>{{lastName}}</td>
        </tr>
        <tr>
            <td>Gender</td>
            <td>{{gender}}</td>
        </tr>
        <tr>
            <td>Qualification</td>
            <td>{{qualification}}</td>
        </tr>
    </tbody>
</table>
```

Output

Student Details	
First Name	Karthik
Last Name	Elumalai
Gender	Male
Qualification	MCA

Negative Test for Attribute Binding

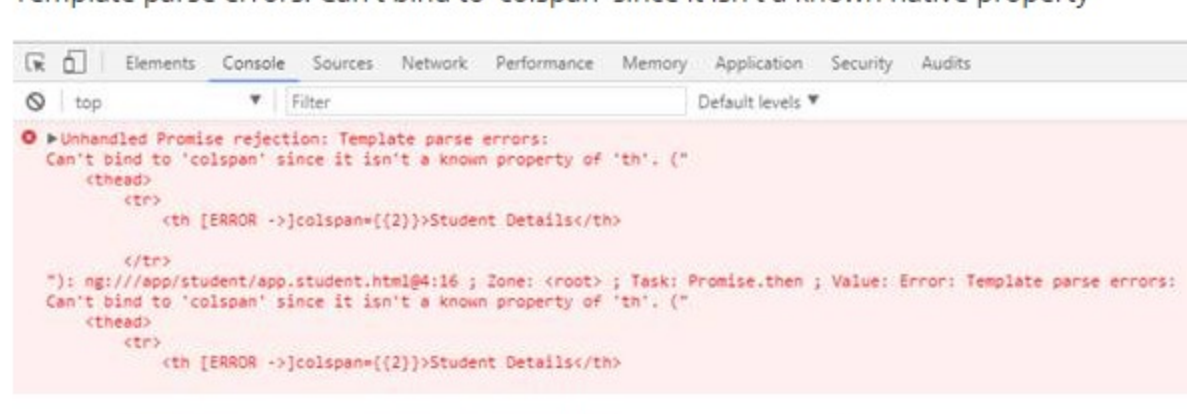
For curiosity, we will check what will be the error if we use interpolation or property binding in the "colspan" attribute. Here is the output.

Ex

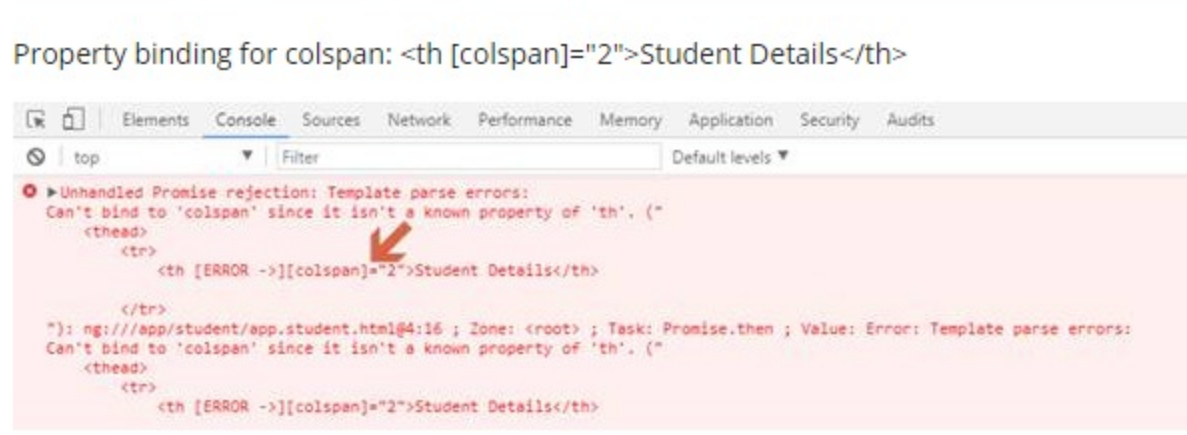
Interpolation for colspan :<th colspan=**{{2}}**>Student Details</th>

Error

Template parse errors: Can't bind to 'colspan' since it isn't a known native property



Property binding for colspan: <th [colspan]="2">Student Details</th>



Class Binding

- By using the class binding, we can add and remove CSS class names from HTML elements
- It is similar to the attribute binding but it starts with the prefix class, optionally followed by a dot (.) and the name of a CSS class: [class. Class-name]

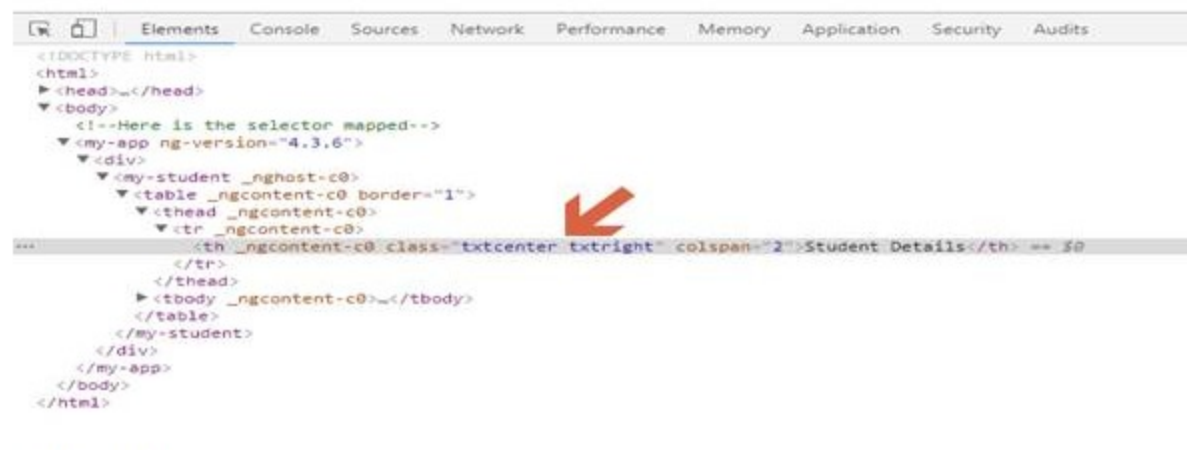
Example of adding a class

```
<style>
    .txtcenter{
    text-align:center;
}

.txtright{
     text-align:right;
}

.txtleft{
       text-align:left;
}
</style>

<table border="1">
    <thead>
        <tr>
            <th [attr.colspan]="2" class="txtcenter"  [class.txtright]='true'>Student Detail

        </tr>
    </thead>
    <tbody>
        <tr>
            <td>First Name</td>
            <td>{{firstName}}</td>
        </tr>
        <tr>
            <td>Last Name</td>
            <td>{{lastName}}</td>
        </tr>
        <tr>
            <td>Gender</td>
            <td>{{gender}}</td>
        </tr>
        <tr>
            <td>Qualification</td>
            <td>{{qualification}}</td>
        </tr>
    </tbody>
</table>
```

In the above code, [class.txtright]='true' added the class txtright for the <th> element.

Output

Student Details	
First Name	Karthik
Last Name	Elumalai
Gender	Male
Qualification	MCA



Style Binding

By using the style binding, we can set inline styles to the HTML element.

One important point here is that it is used to set single line style. If you want to send multiple line style, then Angular provides a good attribute directive called NgStyle.

Syntax

It is similar to the attribute and class binding. It also starts with the prefix style, followed by a dot (.) and the name of a CSS style property: [style. Style-property].

Example

```
<thead>
        <tr>
            <th [attr.colspan]="2"  [style.font-size.px]="50">Student Details</th>

        </tr>
    </thead>
```

Output

Student Details	
First Name	Karthik
Last Name	Elumalai
Gender	Male
Qualification	MCA

Note

This brings us to the end of the first way (Component to View) of one-way data binding. Given below is the second way (View to Component) One-way data binding.

One way Data-Binding [View to Component]

Event Binding

Event binding flows or binds the data from an HTML element to a component.

Syntax

Within parentheses on the left of the equal sign, we have the target event ("click" in this case) and on the right side, we have the template statements such as component properties and methods.

```
<button (click)="onClick()">Click me</button>
```

In this case, the onClick() method of the component class is called when the click event occurs.

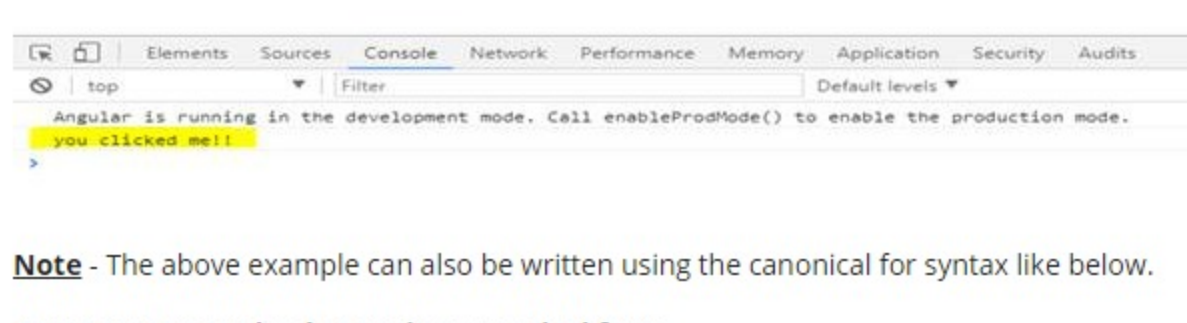
Example

File name app.component.ts

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

@Component({

    selector: 'my-App',
    template: `<button (click)='onClick()' >Click me</button>`

})

export class AppComponent {
    onClick(): void {
        console.log('you clicked me!!');
    }
}
```

Output

Click me



__Note__ - The above example can also be written using the canonical for syntax like below.

Same syntax can be done using canonical form

An event binding using on- keyword is achieved as follows.

```
<button on-click="onClick()">Click me</button>
```

Two-way Data Binding

- In simple words, two-way data binding is a combination of both Property Binding and Event Binding.

```
<input [value]='data1' (input)='data1 = $event.target.value'>
```

- That is, it provides the bi-directional synchronization between the View and the Component.

Example

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

@Component({
    selector: 'my-app',
    template: `
                Enter the value  : <input [value]='data1' (input)='data1 = $event.target.val
                <br>
                 Entered value : Hi  {{data1}}
              `
})
export class AppComponent {
    data1: string = '';
}
```

Output

Enter the value : Karthik
Entered value : Hi Karthik

Binding using [(ngModel)] directive

To simplify the above example, Angular 2 provides the ngModel directive which combines the square brackets of property binding with the parentheses of event binding in a single notation.

Syntax

```
<input [(ngModel)] ='data1'>
```

__**Prerequisite Important Note**__

When you are using NgModule directive, make sure you have imported Angular system Module called Form Module in app.module.ts file like below.

Filename app.module.ts

```
import { FormsModule } from "@angular/forms"

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from "@angular/forms"

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

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

Otherwise, you will get the below error in browser console.

Template parse errors

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

After importing, below is the example which is presented in File Name: app.component.ts

Example

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

@Component({
    selector: 'my-app',
    template: `
                Enter the value  : <input [(ngModel)] ='data1'>
                <br>
                 Entered value : Hi  {{data1}}
              `
})
export class AppComponent {
    data1: string = '';
}
```

Output

Enter the value : Sachin
Entered value : Hi Sachin

Conclusion

In Angular 2, Data Binding is a very good and useful feature, which can be easily implemented in our projects.

Hope, the above information was helpful. Kindly share your thoughts or feedbacks. And, if you like to explore more Angular concepts, here are the links.

- Beginners-guide-for-angular-module
- Angular-component-with-examples
- Angular-directives-with-examples

Angular | One Way Data Bindings | Two Way Data Bindings

Karthik Elumalai TOP 500 338 628.6k
I am Karthik, Software Developer | Microsoft MCSA| Microsoft MCP |Microsoft C# Specialist. I have Hands on Experience in Asp.Net, C#, SQL,JavaScript, CSS, HTML, HTML5, CSS3. I am an engineer by profession and internet su... Read more
http://www.learnandshare-karthik.com/