Property Binding in Angular

2 Comments / 5 minutes of reading / March 9, 2023

← Interpolation | Angular Tutorial | Event Binding →

In this guide let us explore the Property Binding in Angular with examples. Property binding is one way from component to view. It lets you set a property of an element in the view to property in the component. 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`).

Table of Contents

Property Binding Syntax

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.

Property Binding Example

Create a new application

```
ng new property
```

Open `app.component.html`

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

Open the 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;

}
```

We have two property binding in the example above

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` is the component, the Angular automatically updates the view.

Property Binding is one way

Property binding is one way as values go from the component to the template. When the component values change, the Angular updates the view. But if the values changes in the view, the Angular does not update the component.

Should not change the state of the app

The Angular evaluates the template expression (binding-source) to read the values from the component. It then populates the view. If the expression changes any of the component values, then the view would be inconsistent with the model. Hence we need to avoid using expression which will alter the component state.

It means that you cannot make use of the following

- Assignments `(=, +=, -=, ...)`
- Keywords like `new`, `typeof`, `instanceof`, etc
- Chaining expressions with `;` or `,`
- The increment and decrement operators `++` and `--`
- bitwise operators such as `|` and `&`

Return the proper type

The binding expression should return the correct type. The type that the target property expects. Otherwise, it will not work

Property name in camel case

There are few element property names in the camel case, while their corresponding attributes are not. For example `rowSpan` & `colSpan` properties of the table are in the camel case. The HTML attributes are all lowercase (`rowspan` & `colspan`)

Remember the brackets

The brackets, `[]` , tell Angular to evaluate the template expression. If you omit the brackets, Angular treats the expression as a constant string and initializes the target property with that string:

Content Security

Angular inspects the template expression for untrusted values and sanitizes them if found any. For example, the following component variable `evilText` contains the `script` tag. This is what we call the script injection attack. **The Angular does not allow HTML with script tags**. It treats the entire content as string and displays as it is.

```
1
2 Component
3
4 evilText = 'Template <script>alert("You are hacked")</script> Syntax';
5
6 Template
```

```
7
8 <p [textContent]="evilText"></p>
9
```

DOM Properties, not attributes

The property binding binds to the *properties* of DOM elements, components, and directives and not to HTML attributes. The angular has a special syntax for attribute binding.

Special Binding

The Angular has a special syntax for class, styles & attribute binding

The classes & styles are special because they contain a list of classes or styles. The bindings need to be more flexible in managing them. Hence we have a class & style binding.

The Property bindings cover all the properties, but there are certain HTML attributes that do not have any corresponding HTML property. Hence we have attribute 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

Similar to the class, the style also can be set using the following ways. Click on the links to find out more

- Style Property 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  //Template
3
4  //Setting aria label
5  <button [attr.aria-label]="closeLabel" (onclick)="closeMe()">X</button>
6
7  //Table colspan
8  <table border="1">
9    <tr>
10     <td>Col 1</td>
11     <td>Col 2</td>
12     <td>Col 3</td>
13   </tr>
14   <tr>
15      <td [attr.colspan]="2">Col 1 & 2</td>
16      <td>Col 3</td>
```

```
  </tr>
  <tr>
     <td>Col 1</td>
     <td bind-attr.colspan = "getColspan()">Col 2 & 3 </td>
  </tr>
  <tr>
     <td>Col 1</td>
     <td>Col 2</td>
     <td>Col 3</td>
   </tr>

</table>

```

```

//Component

closeLabel="close";
getColspan() {
   return "2"
}

```

Property Binding Vs Interpolation

Everything that can be done from interpolation can also be done using the Property binding. Interpolation is actually a shorthand for binding to the `textContent` property of an element.

For example the following interpolation

```

<h1> {{ title }} </h1>

```

Is same as the following Property binding

```

```

```
<h1 [innerText]="title"></h1>
```

In fact, Angular automatically translates interpolations into the corresponding property bindings before rendering the view.

Interpolation is simple and readable. For example, the above example of setting the `h1` tag, the in interpolation is intuitive and readable than the property binding syntax

Interpolation requires the expression to return a string. If you want to set an element property to a non-string data value, you must use property binding.

Property Binding Example

Binding to innerHTML with HTML tags

Here the Angular parses the `b` & `p` tags and renders it in the view.

```

//Template
<p [innerHTML]="text1"></p>
<div [innerHTML]="text2"></div>

```

```
//Component
 text1="The <b>Angular</b> is printed in bold"
text2="<p>This is first para</p><p>This is second para</p> "
```

img

```
//Template

<img [src]="itemImageUrl">
<img bind-src="itemImageUrl">
```

```
//Component
itemImageUrl="https://angular.io/assets/images/logos/angular/logo-nav@2x.png"
```

Concatenate two string

```
<p [innerText]="'Hello & Welcome to '+ ' Angular Data binding '"></p>
```

Mathematical expressions

```
<p [innerText]="100*80"></p>
```

setting the color

```
//template
```