Property binding

Property binding in Angular helps you set values for properties of HTML elements or directives. Use property binding to do things such as toggle button functionality, set paths programmatically, and share values between components.

> See the live example / download example for a working example containing the code snippets in this guide.

Prerequisites

- Basics of components
- Basics of templates
- Binding syntax

Understanding the flow of data

Property binding moves a value in one direction, from a component's property into a target element property.

> For more information on listening for events, see Event binding.

To read a target element property or call one of its methods, see the API reference for ViewChild and ContentChild.

Binding to a property

To bind to an element's property, enclose it in square brackets, `[]`, which identifies the property as a target property.

A target property is the DOM property to which you want to assign a value.

To assign a value to a target property for the image element's `src` property, type the following code:

src/app/app.component.html

```
<img alt="item" [src]="itemImageUrl">
```

In most cases, the target name is the name of a property, even when it appears to be the name of an attribute.

In this example, `src` is the name of the `<img>` element property.

The brackets, `[]`, cause Angular to evaluate the right-hand side of the assignment as a dynamic expression.

Without the brackets, Angular treats the right-hand side as a string literal and sets the property to that static value.

To assign a string to a property, type the following code:

src/app.component.html

```
<app-item-detail childItem="parentItem"></app-item-detail>
```

Omitting the brackets renders the string `parentItem`, not the value of `parentItem`.

Setting an element property to a component property value

To bind the `src` property of an `<img>` element to a component's property, place the target, `src`, in square brackets followed by an equal sign and then the property.

Using the property `itemImageUrl`, type the following code:

src/app/app.component.html

```
<img alt="item" [src]="itemImageUrl">
```

Declare the `itemImageUrl` property in the class, in this case `AppComponent`.

src/app/app.component.ts

```
itemImageUrl = '../assets/phone.png';
```

`colspan` and `colSpan`

A common point of confusion is between the attribute, `colspan`, and the property, `colSpan`. Notice that these two names differ by only a single letter.

To use property binding using colSpan, type the following:

src/app/app.component.html

```
<!-- Notice the colSpan property is camel case -->
<tr><td [colSpan]="1 + 1">Three-Four</td></tr>
```

To disable a button when the component says that it `isUnchanged`, type the following:

src/app/app.component.html

```
<!-- Bind button disabled state to `isUnchanged`
property -->
<button type="button" [disabled]="isUnchanged">Disabled
Button</button>
```

To set a property of a directive, type the following:

src/app/app.component.html

```
<p [ngClass]="classes">[ngClass] binding to the classes
property making this blue</p>
```

To set the model property of a custom component for parent and child components to communicated, type the following:

src/app/app.component.html

```
<app-item-detail [childItem]="parentItem"></app-item-
detail>
```

Toggling button functionality

To disable a button's functionality depending on a Boolean value, bind the DOM `disabled` property to a property in the class that is `true` or `false`.

src/app/app.component.html

```html
<!-- Bind button disabled state to `isUnchanged` property -->
<button type="button" [disabled]="isUnchanged">Disabled Button</button>
```

Because the value of the property `isUnchanged` is `true` in the `AppComponent`, Angular disables the button.

src/app/app.component.ts

```typescript
isUnchanged = true;
```

What's next

- Property binding best practices
- Event binding
- Text Interpolation
- Class & Style Binding
- Attribute Binding

Last reviewed on Thu Apr 14 2022