Property binding best practices

By following a few guidelines, you can use property binding in a way that helps you minimize bugs and keep your code readable.

See the for a working example containing the code snippets in this guide.

Avoid side effects

Evaluation of a template expression should have no visible side effects. Use the syntax for template expressions to help avoid side effects. In general, the correct syntax prevents you from assigning a value to anything in a property binding expression. The syntax also prevents you from using increment and decrement operators.

An example of producing side effects

If you had an expression that changed the value of something else that you were binding to, that change of value would be a side effect. Angular might or might not display the changed value. If Angular does detect the change, it throws an error.

As a best practice, use only properties and methods that return values.

Return the proper type

A template expression should evaluate to the type of value that the target property expects. For example, return a string if the target property expects a string, a number if it expects a number, or an object if it expects an object.

Passing in a string

In the following example, the `childItem` property of the `ItemDetailComponent` expects a string.

Confirm this expectation by looking in the `ItemDetailComponent` where the `@Input()` type is `string`:

The `parentItem` in `AppComponent` is a string, which means that the expression, `parentItem` within `[childItem]="parentItem"`, evaluates to a string.

If `parentItem` were some other type, you would need to specify `childItem` `@Input()` as that type as well.

Passing in an object

In this example, `ItemListComponent` is a child component of `AppComponent` and the `items` property expects an array of objects.

In the `ItemListComponent` the `@Input()`, `items`, has a type of `Item[]`.

Notice that `Item` is an object that it has two properties; an `id` and a `name`.

In `app.component.ts`, `currentItems` is an array of objects in the same shape as the `Item` object in `items.ts`, with an `id` and a `name`.

By supplying an object in the same shape, you satisfy the expectations of `items` when Angular evaluates the expression `currentItems`.