Two-way binding `[(...)]`

Contents

Two-way binding gives your app a way to share data between a component class and its template.

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

Basics of two-way binding

Two-way binding does two things:

1. Sets a specific element property.
2. Listens for an element change event.

Angular offers a special *two-way data binding* syntax for this purpose, `[()]`. The `[()]` syntax combines the brackets of property binding, `[]`, with the parentheses of event binding, `()`.

[()] = BANANA IN A BOX

Visualize a *banana in a box* to remember that the parentheses go *inside* the brackets.

The `[()]` syntax is easy to demonstrate when the element has a settable property called `x` and a corresponding event named `xChange`. Here's a `SizerComponent` that fits this pattern. It has a `size` value property and a companion `sizeChange` event:

src/app/sizer.component.ts

```
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-sizer',
  templateUrl: './sizer.component.html',
  styleUrls: ['./sizer.component.css']
})
```

```
export class SizerComponent {


  @Input()  size: number | string;
  @Output() sizeChange = new EventEmitter<number>();

  dec() { this.resize(-1); }
  inc() { this.resize(+1); }

  resize(delta: number) {
    this.size = Math.min(40, Math.max(8, +this.size + delta));
    this.sizeChange.emit(this.size);
  }

}
```

src/app/sizer.component.html

```
<div>
  <button (click)="dec()" title="smaller">-</button>
  <button (click)="inc()" title="bigger">+</button>
  <label [style.font-size.px]="size">FontSize: {{size}}px</label>
</div>
```

The initial `size` is an input value from a property binding. Clicking the buttons increases or decreases the `size`, within min/max value constraints, and then raises, or emits, the `sizeChange` event with the adjusted size.

Here's an example in which the `AppComponent.fontSizePx` is two-way bound to the `SizerComponent`:

src/app/app.component.html (two-way-1)

```
<app-sizer [(size)]="fontSizePx"></app-sizer>
<div [style.font-size.px]="fontSizePx">Resizable Text</div>
```

The `AppComponent.fontSizePx` establishes the initial `SizerComponent.size` value.

src/app/app.component.ts

```
fontSizePx = 16;
```

Clicking the buttons updates the `AppComponent.fontSizePx` via the two-way binding. The revised `AppComponent.fontSizePx` value flows through to the *style* binding, making the displayed text bigger or smaller.

The two-way binding syntax is really just syntactic sugar for a *property* binding and an *event* binding. Angular desugars the `SizerComponent` binding into this:

src/app/app.component.html (two-way-2)

```
<app-sizer [size]="fontSizePx" (sizeChange)="fontSizePx=$event"></app-sizer>
```

The `$event` variable contains the payload of the `SizerComponent.sizeChange` event. Angular assigns the `$event` value to the `AppComponent.fontSizePx` when the user clicks the buttons.

Two-way binding in forms

The two-way binding syntax is a great convenience compared to separate property and event bindings. It would be convenient to use two-way binding with HTML form elements like `<input>` and `<select>`. However, no native HTML element follows the `x` value and `xChange` event pattern.

For more on how to use two-way binding in forms, see Angular NgModel.

Version 10.0.10-local+sha.b32126c335.