Template statements

Contents

A template **statement** responds to an **event** raised by a binding target such as an element, component, or directive.

> See the Template syntax / download example for the syntax and code snippets in this guide.

The following template statement appears in quotes to the right of the `=` symbol as in `(event)="statement"`.

src/app/app.component.html

```
<button (click)="deleteHero()">Delete hero</button>
```

A template statement *has a side effect*. That's the whole point of an event. It's how you update application state from user action.

Responding to events is the other side of Angular's "unidirectional data flow". You're free to change anything, anywhere, during this turn of the event loop.

Like template expressions, template *statements* use a language that looks like JavaScript. The template statement parser differs from the template expression parser and specifically supports both basic assignment (=) and chaining expressions with `;`.

However, certain JavaScript and template expression syntax is not allowed:

- `new`
- increment and decrement operators, `++` and `--`
- operator assignment, such as `+=` and `-=`
- the bitwise operators, such as `|` and `&`
- the pipe operator

Statement context

As with expressions, statements can refer only to what's in the statement context such as an event handling method of the component instance.

The *statement context* is typically the component instance. The *deleteHero* in `(click)="deleteHero()"` is a method of the data-bound component.

src/app/app.component.html

```
<button (click)="deleteHero()">Delete hero</button>
```

The statement context may also refer to properties of the template's own context. In the following examples, the template `$event` object, a template input variable (`let hero`), and a template reference variable (`#heroForm`) are passed to an event handling method of the component.

src/app/app.component.html

```
<button (click)="onSave($event)">Save</button>
<button *ngFor="let hero of heroes" (click)="deleteHero(hero)">{{hero.name}}</button>
<form #heroForm (ngSubmit)="onSubmit(heroForm)"> ... </form>
```

Template context names take precedence over component context names. In `deleteHero(hero)` above, the `hero` is the template input variable, not the component's `hero` property.

Statement guidelines

Template statements cannot refer to anything in the global namespace. They can't refer to `window` or `document`. They can't call `console.log` or `Math.max`.

As with expressions, avoid writing complex template statements. A method call or simple property assignment should be the norm.