Template statements

Template statements are methods or properties that you can use in your HTML to respond to user events. With template statements, your application can engage users through actions such as displaying dynamic content or submitting forms.

See the Template syntax for the syntax and code snippets in this guide.

In the following example, the template statement `deleteHero()` appears in quotes to the right of the `=` symbol as in `(event)="statement"`.

When the user clicks the **Delete hero** button, Angular calls the `deleteHero()` method in the component class.

Use template statements with elements, components, or directives in response to events.

Responding to events is an aspect of Angular's unidirectional data flow. You can change anything in your application during a single event loop.

Syntax

Like template expressions, template statements use a language that looks like JavaScript. However, the parser for template statements differs from the parser for template expressions. In addition, the template statements parser specifically supports both basic assignment, `=`, and chaining expressions with semicolons, `;`.

The following 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

Statements have a context—a particular part of the application to which the statement belongs.

Statements can refer only to what's in the statement context, which is typically the component instance. For example, `deleteHero()` of `(click)="deleteHero()"` is a method of the component in the following snippet.

The statement context may also refer to properties of the template's own context. In the following example, the component's event handling method, `onSave()` takes the template's own `$event` object as an argument. On the next two lines, the `deleteHero()` method takes a template input variable, `hero`, and `onSubmit()` takes a template reference variable, `#heroForm`.

In this example, the context of the `$event` object, `hero`, and `#heroForm` is the template.

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

Statement best practices

- **Conciseness**

Use method calls or basic property assignments to keep template statements minimal.

- **Work within the context**

  The context of a template statement can be the component class instance or the template. Because of this, template statements cannot refer to anything in the global namespace such as `window` or `document`. For example, template statements can't call `console.log()` or `Math.max()`.