Event binding

Event binding lets you listen for and respond to user actions such as keystrokes, mouse movements, clicks, and touches.

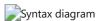
See the for a working example containing the code snippets in this guide.

Binding to events

To bind to an event you use the Angular event binding syntax. This syntax consists of a target event name within parentheses to the left of an equal sign, and a quoted template statement to the right. In the following example, the target event name is click and the template statement is onSave().

```
<button (click)="onSave()">Save</button>
```

The event binding listens for the button's click events and calls the component's onSave () method whenever a click occurs.



Binding to passive events

Angular also supports passive event listeners. For example, use the following steps to make a scroll event passive.

- 1. Create a file zone-flags.ts under src directory.
- 2. Add the following line into this file.

```
(window as any)['__zone_symbol__PASSIVE_EVENTS'] = ['scroll'];
```

3. In the src/polyfills.ts file, before importing zone.js, import the newly created zone-flags .

```
import './zone-flags';
import 'zone.js'; // Included with Angular CLI.
```

After those steps, if you add event listeners for the <code>scroll</code> event, the listeners will be <code>passive</code> .

Custom events with EventEmitter

<u>Directives</u> typically raise custom events with an Angular <u>EventEmitter</u> as follows.

- 1. The directive creates an EventEmitter and exposes it as a property.
- 2. The directive then calls EventEmitter.emit(data) to emit an event, passing in message data, which can be anything.
- 3. Parent directives listen for the event by binding to this property and accessing the data through the \$event object.

Consider an ItemDetailComponent that presents item information and responds to user actions. Although the ItemDetailComponent has a delete button, it doesn't contain the functionality to delete the hero. It can only raise an event reporting the user's delete request.

The component defines a deleteRequest property that returns an EventEmitter . When the user clicks **Delete**, the component invokes the delete() method, telling the EventEmitter to emit an Item object. The hosting parent component binds to the deleteRequest event of the ItemDetailComponent as follows.

When the deleteRequest event fires, Angular calls the parent component's deleteItem() method with the item.

Determining an event target

To determine an event target, Angular checks if the name of the target event matches an event property of a known directive. In the following example, Angular checks to see if myClick is an event on the custom
ClickDirective.

If the target event name, myClick fails to match an element event or an output property of ClickDirective, Angular reports an "unknown directive" error.

What's next

For more information on how event binding works, see **How event binding works**.