Enlace de eventos

La vinculación de eventos le permite escuchar y responder a las acciones del usuario, como pulsaciones de teclas, movimientos del mouse, clics y toques.

Ver el [ejemplo en vivo](https://angular.io/generated/live-examples/event-binding/stackblitz.html)/ [ejemplo de descarga](https://angular.io/generated/zips/event-binding/event-binding.zip) para ver un ejemplo funcional que contiene los fragmentos de código de esta guía.

Vinculante a eventos

Para enlazar a un evento, usa la sintaxis de enlace de eventos Angular. Esta sintaxis consta de un nombre de evento de destino entre paréntesis a la izquierda de un signo igual y una declaración de plantilla entre comillas a la derecha. En el siguiente ejemplo, el nombre del evento de destino es click y la declaración de plantilla es onSave().

Sintaxis de enlace de eventos

content\_copy<button (click)="onSave()">Save</button>

El enlace de eventos escucha los eventos de clic del botón y llama al onSave()método del componente cada vez que se produce un clic.

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

El enlace de eventos escucha los eventos de clic del botón y llama al onSave()método del componente cada vez que se produce un clic



Vinculación a eventos pasivos

Angular también admite detectores de eventos pasivos. Por ejemplo, utilice los siguientes pasos para convertir un evento de desplazamiento en pasivo.

1. Cree un archivo zone-flags.tsen el srcdirectorio.
2. Agregue la siguiente línea a este archivo.

(window as any)['\_\_zone\_symbol\_\_PASSIVE\_EVENTS'] = ['scroll'];

1. En el src/polyfills.tsarchivo, antes de importar zone.js, importe el archivo zone-flags.

content\_copyimport './zone-flags';

import 'zone.js'; // Included with Angular CLI.

Después de esos pasos, si agrega detectores de eventos para el scrollevento, los detectores serán passive.

Eventos personalizados con [EventEmitter](https://angular.io/api/core/EventEmitter)

[Las directivas](https://angular.io/guide/built-in-directives) suelen generar eventos personalizados con un Angular [EventEmitter de la](https://angular.io/api/core/EventEmitter) siguiente manera.

1. La directiva crea un [EventEmitter](https://angular.io/api/core/EventEmitter)y lo expone como una propiedad.
2. Luego, la directiva llama EventEmitter.emit(data)para emitir un evento, pasando datos del mensaje, que pueden ser cualquier cosa.
3. Las directivas principales escuchan el evento enlazándose a esta propiedad y accediendo a los datos a través del $event objeto.

Considere un ItemDetailComponentque presenta información sobre el artículo y responde a las acciones del usuario. Aunque ItemDetailComponenttiene un botón de eliminación, no contiene la funcionalidad para eliminar al héroe. Solo puede generar un evento que informe la solicitud de eliminación del usuario.

src / app / item-detail / item-detail.component.html (plantilla)

content\_copy<img src="{{itemImageUrl}}" [style.display]="displayNone">

<span [style.text-decoration]="lineThrough">{{ item.name }}

</span>

<button (click)="delete()">Delete</button>

El componente define una deleteRequestpropiedad que devuelve un [EventEmitter](https://angular.io/api/core/EventEmitter). Cuando el usuario hace clic en Eliminar , el componente invoca el delete()método y le dice [EventEmitter](https://angular.io/api/core/EventEmitter)que emita un Itemobjeto.

src / app / item-detail / item-detail.component.ts (deleteRequest)

content\_copy// This component makes a request but it can't actually delete a hero.

@[Output](https://angular.io/api/core/Output)() deleteRequest = new [EventEmitter](https://angular.io/api/core/EventEmitter)<Item>();

delete() {

this.deleteRequest.emit(this.item);

this.displayNone = this.displayNone ? '' : 'none';

this.lineThrough = this.lineThrough ? '' : 'line-through';

}

El componente principal de alojamiento se une al deleteRequestevento de la ItemDetailComponentsiguiente manera.

src / app / app.component.html (enlace de evento a componente)

content\_copy<app-item-detail (deleteRequest)="deleteItem($event)" [item]="currentItem"></app-item-detail>

Cuando se deleteRequestactiva el evento, Angular llama al deleteItem()método del componente principal con el elemento.

Determinar un objetivo de evento

Para determinar un objetivo de evento, Angular comprueba si el nombre del evento de destino coincide con una propiedad de evento de una directiva conocida. En el siguiente ejemplo, Angular comprueba si myClickhay un evento en el archivo personalizado ClickDirective.

src / app / app.component.html

content\_copy<h4>myClick is an event on the custom ClickDirective:</h4>

<button (myClick)="clickMessage=$event" clickable>click with myClick</button>

{{clickMessage}}

Si el nombre del evento de destino myClick no coincide con un evento de elemento o una propiedad de salida de ClickDirective, Angular informa un error de "directiva desconocida".

Una forma común de manejar eventos es pasar el objeto de evento $event, al método que maneja el evento. El $eventobjeto a menudo contiene información que el método necesita, como el nombre de un usuario o la URL de una imagen.

El evento de destino determina la forma del $eventobjeto. Si el evento de destino es un evento de elemento DOM nativo, entonces $eventes un[Objeto de evento DOM](https://developer.mozilla.org/en-US/docs/Web/Events), con propiedades como targety target.value.

En el siguiente ejemplo, el código establece la <input> valuepropiedad vinculando a la namepropiedad.

src / app / app.component.html

content\_copy<input [value]="currentItem.name"

(input)="currentItem.name=getValue($event)">

Con este ejemplo, ocurren las siguientes acciones:

1. El código se une al inputevento del <input>elemento, lo que permite que el código escuche los cambios.
2. Cuando el usuario realiza cambios, el componente genera el inputevento.
3. Se ejecuta la vinculación de la declaración dentro de un contexto que incluye el objeto de evento DOM, $event.
4. Angular recupera el texto modificado llamando getValue($event.target)y actualiza la namepropiedad.

Si el evento pertenece a una directiva o componente, $eventtiene la forma que produce la directiva o el componente.

El tipo de $event.targetestá solo EventTargeten la plantilla. En el getValue()método, el destino se convierte en an HTMLInputElementpara permitir el acceso con seguridad de tipos a su valuepropiedad.

|  |
| --- |
| getValue(event: Event): string {  return (event.target as HTMLInputElement).value;  } |

## Eventos tipos

(focus)="myMethod()"

(blur)="myMethod()"

(submit)="myMethod()"

(scroll)="myMethod()"

(cut)="myMethod()"

(copy)="myMethod()"

(paste)="myMethod()"

(keydown)="myMethod()"

(keypress)="myMethod()"

(keyup)="myMethod()"

(mouseenter)="myMethod()"

(mousedown)="myMethod()"

(mouseup)="myMethod()"

(click)="myMethod()"

(dblclick)="myMethod()"

(drag)="myMethod()"

(dragover)="myMethod()"

(drop)="myMethod()"