



flex, grid...

CTRL+K

# El método addEventListener

Gestionar eventos Javascript me|

En los artículos anteriores hemos visto que son los **eventos Javascript** y como gestionarlos a través de código HTML, o a través de código Javascript, utilizando la API del DOM. Sin embargo, la forma más recomendable es hacer uso del método `.addEventListener()`, el cuál es mucho más potente y versátil para la mayoría de los casos.

- Con `.addEventListener()` se pueden añadir fácilmente **múltiples** funcionalidades.
- Con `.removeEventListener()` se puede **eliminar** una funcionalidad previamente añadida.
- Con `.addEventListener()` se pueden indicar ciertos **comportamientos** especiales.

## Método `.addEventListener()`

Con el método `.addEventListener()` permite añadir una escucha del **evento** indicado ( **primer parámetro** ), y en el caso de que ocurra, se ejecutará la función asociada indicada ( **segundo parámetro** ). De forma opcional, se le puede pasar un tercer parámetro **OBJECT** con ciertas opciones, que veremos más adelante:

Método
Descripción
<code>.addEventListener( <b>S</b> event, <b>F</b> func )</code>
Escucha el evento <code>event</code> , y si ocurre, ejecuta <code>func</code> .
<code>.addEventListener( <b>S</b> event, <b>F</b> func, <b>O</b> options )</code>
Idem, pasándole ciertas opciones.

Para verlo en acción, vamos a crear a continuación, el mismo ejemplo de apartados anteriores, de esta forma veremos como funciona y podremos comparar con los anteriores:

JS

HTML

DEMO

```
const button = document.querySelector("button");

function action() {
  alert("Hello!");
};

button.addEventListener("click", action);
```

Observa detenidamente lo que hacemos en este ejemplo:

- **1** Buscamos el elemento que tendrá el evento, en este caso `<button>`
- **2** Creamos una función `action()` que realizará la acción deseada.
- **3** En el botón, escuchamos el evento `click` y le asociamos la función `action`.

Ten muy en cuenta los siguientes detalles:

- ✨ En el **primer parámetro** indicamos el nombre del evento, en nuestro ejemplo, `click`.
- ✨ Con `.addEventListener()` no se precede con `on` y se escriben en minúsculas, sin camelCase.
- ✨ En el **segundo parámetro** indicamos la función a ejecutar al ocurrir el evento.

[addEventListener](#)   [Múltiples listeners](#)   [Opciones](#)

## ↳ Detalles sobre `.addEventListener()`

Aunque es posible que al principio veas más organizado este código, es muy habitual escribir los eventos de esta otra forma:

```
const button = document.querySelector("button");

button.addEventListener("click", function() {
  alert("Hello!");
});
```

JS

En lugar de tener la función definida fuera, la utilizamos en el propio `.addEventListener()` y nos ahorramos ponerle un nombre, es decir, utilizamos una **función anónima**. Si prefieres utilizar las [funciones flecha](#) de Javascript, quedaría incluso más legible:

JS

```
const button = document.querySelector("button");  
button.addEventListener("click", () => alert("Hello!"));
```

Una de las características más cómodas de utilizar `.addEventListener()` es que puedes añadir múltiples listeners de forma sencilla, así que lo veremos a continuación.

## Método `.removeEventListener()`

El ejemplo anterior, se puede completar haciendo uso del método `.removeEventListener()`, que sirve como su propio nombre indica para eliminar un listener que se ha añadido previamente al elemento. Para ello es muy importante indicar **la misma función** que añadimos con el `.addEventListener()` y no una **función diferente que haga lo mismo que la primera**.

Método
Descripción
<code>.removeEventListener( s event, f func)</code>
Elimina la funcionalidad <code>func</code> asociada al evento <code>event</code> .

Veamos el ejemplo anterior, eliminando la funcionalidad `action` mediante `.removeEventListener()`, es decir, sólo debería actuar la funcionalidad `toggle`:

JS CSS HTML DEMO

```
const button = document.querySelector("button");  
const action = () => alert("Hello!");  
const toggle = () => button.classList.toggle("red");  
  
button.addEventListener("click", action); // Add listener  
button.addEventListener("click", toggle); // Toggle red CSS  
button.removeEventListener("click", action); // Delete listener
```

Ten en cuenta que es posible eliminar el listener del evento porque hemos guardado en una constante la función, y tanto en `.addEventListener()` como en `.removeEventListener()` estamos haciendo referencia a la misma función. Si en lugar de esto, añadiéramos la función literalmente, aunque hagan lo mismo, serían funciones diferentes y no realizaría lo que esperamos.



**Eventos mediante Javascript**  
Capítulo anterior

**Escuchar eventos y handleEvent**  
Capítulo siguiente

**Volver**  
Al índice

**Acceder a Discord**  
Comunidad de Manz.dev

## RELACIONADOS

En mis canales de Youtube [@ManzDev](#) y [ManzDevTy](#), tienes más contenido...




## APRENDER MÁS

Si lo prefieres, puedes aprender también sobre estas temáticas:



## ¿QUIÉN SOY YO?

Soy Manz, vivo en Tenerife (España) y soy streamer partner en Twitch  y profesor. Me apasiona el universo de la programación web, el diseño y desarrollo web y la tecnología en general. Aunque soy full-stack, mi pasión es el front-end, la terminal y crear cosas divertidas y locas.

Puedes encontrar más sobre mi en [Manz.dev](https://manz.dev)



Twitch



Youtube



Twitter



Instagram



Tiktok



Linkedin



GitHub



Discord

Creado por Manz con  Alojado en [DigitalOcean](https://digitalocean.com).  
© Todos los derechos reservados. Los izquierdos también.