



flex, grid...

CTRL+K

El objeto Event

Cómo gestionar los eventos Javascript|

Cuando se disparan ciertos eventos, hay casos en los que nos podría interesar obtener **información relacionada** con la naturaleza del evento en cuestión. Por ejemplo, si estamos escuchando un evento de tipo `click` de ratón, nos podría interesar saber **con que botón** del ratón se ha hecho click, **en que punto concreto** de la pantalla se ha hecho click, etc.

El objeto Event

Estos detalles se pueden obtener de forma opcional a través de un objeto `event` que se proporciona en la función asociada al evento. Para ello, sólo es necesario indicar un **primer parámetro** en la función que gestiona el evento, y dicho parámetro, será de tipo evento con dicha información asociada.

Observa el siguiente ejemplo de código:

JS

```
const button = document.querySelector("button");
button.addEventListener("click", (event) => {
  console.log(event);
});
```

Simplemente se trata de una función que escucha los eventos de tipo `click` en un `<button>` del HTML. Observa que la función asociada tiene el parámetro `event`. Si hacemos click en el botón, en la consola se nos mostrará la información de este evento, que en nuestro ejemplo anterior está basado en `PointerEvent`, ya que `click` es un evento realizado con un puntero (generalmente, de ratón).

Dicho evento contiene una serie de propiedades interesantes a la hora de trabajar con el evento en cuestión, y dependen del tipo de evento. Por ejemplo, en el ejemplo anterior hemos gestionado un evento `click`, por lo que el tipo de evento asociado es `PointerEvent`, y conlleva una serie de propiedades que no tienen porque estar presentes en otros tipos de eventos.

En nuestro ejemplo anterior, tendrían algo parecido a lo siguiente:

```
// Objeto PointerEvent
{
  type: "click",           // Nombre del evento
  pointerType: "mouse"    // Tipo de dispositivo
  altKey: false,          // ¿La tecla ALT estaba presionada?
  ctrlKey: false,         // ¿La tecla CTRL estaba presionada?
  shiftKey: false,        // ¿La tecla SHIFT estaba presionada?
  target: button,         // Referencia al elemento que disparó el evento
  clientX: 43,            // Posición en eje X donde se hizo click
  clientY: 16,            // Posición en eje Y donde se hizo click
  detail: 1,              // Contador de veces que se ha hecho click
  path: [],               // Camino por donde ha pasado el evento
  ...                     // Otros...
}
```

Ten en cuenta que en este caso, se trata de un objeto `PointerEvent` porque el evento que estamos escuchando es un evento `click` de un dispositivo que permite apuntar. Sin embargo, si utilizáramos otro evento, posiblemente obtendríamos un objeto con propiedades diferentes.



Propiedades del evento

Veamos algunas de las propiedades comunes, que están disponibles en cualquier tipo de evento. Ampliemos un poco la escucha del evento anterior, donde vamos a observar el contenido de cada una de las siguientes propiedades básicas:

```
const button = document.querySelector("button");
button.addEventListener("click", (event) => {
  const { type, timeStamp, isTrusted } = event;
  console.log({ type, timeStamp, isTrusted });
});
```

Como se puede ver, desestructuramos las tres propiedades siguientes del objeto `event` y las mostramos a través de una sentencia `console.log()`, de modo que podamos ver su contenido.

Propiedad
Descripción
<code>event.type</code>
Indica el tipo de evento en cuestión.

Propiedad
Descripción
<code>N .timeStamp</code>
Hora en milisegundos en la que se creó el evento.
<code>B .isTrusted</code>
Indica si es un evento real de un usuario o uno enviado manualmente con <code>.dispatchEvent()</code> .

Analicemos cada una de ellas.

`type` `timeStamp` `isTrusted`

↳ Tipo de evento

Mediante la propiedad `.type` podemos obtener el **tipo de evento**. Esto es, simplemente, el nombre del evento con el que escuchamos en el `.addEventListener()`, o en el caso de un [evento personalizado](#), el establecido en el primer parámetro de la instancia del `new CustomEvent()`.

En el ejemplo inicial de este artículo estábamos escuchando un evento nativo donde `type` sería `click`.

Evitar la acción por defecto

Algunos elementos tienen un **comportamiento por defecto**. Por ejemplo, el elemento `<details>` muestra el texto del elemento `<summary>`, si se pulsa sobre el, se despliega el resto del contenido de `<details>`. Si se vuelve a pulsar, se oculta nuevamente. Ese es su **comportamiento por defecto**.

Sin embargo, pueden existir situaciones donde queremos que se anule este comportamiento y no se realice. Por ejemplo, para reimplementarlo nosotros, o cambiar su funcionalidad habitual. Para ello, tenemos a nuestra disposición una propiedad y un método que harán que sea muy sencillo:

Propiedad o Método
Valor por defecto
Descripción
Propiedad
<code>B .defaultPrevented</code>

Propiedad o Método
Valor por defecto
Descripción
false
Indica si el comportamiento por defecto se ha evitado.
Métodos
.preventDefault()
Evita que se realice el comportamiento por defecto del evento.

Mediante el método `.preventDefault()` se establecerá el flag `.defaultPrevented` a `true` y podremos evitar el comportamiento base por defecto de dicho evento y añadirle otro diferente:

JS HTML DEMO

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

details.addEventListener("click", (event) => {
  event.preventDefault();
  alert("Has hecho click pero hemos eliminado el comportamiento por defecto.");
});
```

De esta forma, veremos que al pulsar sobre el elemento `<details>` ya no se expande ni se contrae, por lo que ahora podríamos crear nuestra propia lógica para reimplementar esta funcionalidad.

Escuchar eventos y handleEvent
Capítulo anterior

¿Qué son los Custom Events?
Capítulo siguiente

Volver
Al índice

Acceder a Discord
Comunidad de Manz.dev

RELACIONADOS

En mis canales de Youtube [@ManzDev](#) y [ManzDevTv](#), 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](#).
© Todos los derechos reservados. Los izquerdos también.