Eventos

Ángel Villalba Fdez-Paniagua

Table of Contents

1. Introducción

Cuando estamos navegando por internet, el navegador emite distintos tipos de eventos (de ratón, de teclado…), que es la forma que tiene de decirnos lo que está ocurriendo.

Los **eventos** se van creando según los usuarios interactuan con el navegador (pulsan un botón, escriben algo en un campo de texto…). Cuando estos eventos emiten, se pueden detectar para ejecutar funciones que tenemos en JavaScript. Aunque el evento sea el mismo, puede variar la función que se va a ejecutar dependiendo de que elemento sea el que haya detectado el evento. Estas funciones que se ejecutan, suelen dar alguna respuesta, por ejemplo actualizando el contenido de la página.

2. Tipos de eventos

Los eventos que el navegador emite se pueden dividir en distintos tipos, los que emite el teclado, el ratón, un formulario… Y estos eventos se suelen usar para ejecutar una función con código JavaScript cuando se emiten.

Table 1. Eventos de la interfaz

Evento	Descripción
load	Cuando la página se ha terminado de cargar
resize	Cuando se cambia el tamaño de la ventana del navegador
scroll	Cuando se hace scroll en la página

Table 2. Eventos de teclado

Evento	Descripción
keydown	El usuario pulsa cualquier tecla. Se emite el evento mientras la tecla está pulsada
keyup	El usuario deja de pulsar la tecla
keypress	El usuario pulsa una tecla (letras, números, caracteres especiales, espacio y enter). Se emite el evento mientras la tecla está pulsada

Table 3. Eventos de ratón

Evento	Descripción
click	El usuario pulsa sobre un elemento
dblclick	El usuario pulsa sobre un elemento dos veces
mousedown	El usuario pulsa el ratón mientras está sobre el elemento
mouseup	El usuario deja de pulsar el ratón mientras está sobre el elemento. Si deja de pulsarlo en otro elemento distinto, no se emite el evento

Evento	Descripción
mousemove	Cuando el usuario mueve el ratón
mouseover	Cuando el usuario mueve el ratón sobre un elemento
mouseout	Cuando el usuario mueve el ratón fuera del elemento

Table 4. Eventos de foco

Evento	Descripción
focus/focusin	Cuando el elemento obtiene el foco
blur/focusout	Cuando el elemento pierde el foco

Table 5. Eventos de formularios

Evento	Descripción
input	Mientras cambia el valor en cualquier elemento `input` o `textarea`
change	Cuando cambia el valor de algún campo del formulario
submit	Cuando se envía el formulario (usando un botón de tipo `submit` o pulsando la tecla `enter`)
reset	Cuando se resetea el formulario
cut	Cuando el usuario corta el contenido de algún campo del formulario
copy	Cuando el usuario copia el contenido de algún campo del formulario
paste	Cuando el usuario pega contenido en algún campo del formulario
select	Cuando el usuario selecciona texto en algún campo del formulario

3. Manejadores de eventos

Cuando el usuario interactua con la página, ocurren una serie de pasos a los cuales vamos a llamar **manejadores de eventos** o **event handlers**. Estos pasos son:

- Se selecciona el elemento sobre el que se quiere detectar un evento
- Se indica que tipo de evento se quiere detectar
- Se establece el código que se va a ejecutar cuando ocurra el evento sobre ese elemento

Por ejemplo, si queremos mostrar un mensaje por consola cuando se pulse sobre un botón, tendríamos que seguir los siguientes pasos. Primero obtenemos el botón usando los métodos de búsqueda del DOM. Después le asignamos el evento `click` a ese botón que hemos obtenido. Y por último, al asignar el evento se le indica que código vamos a ejecutar, en este caso se le dirá que

ejecute la función `saludar()`.

Podemos manejar estos eventos de tres formas distintas, es decir, que podemos diferenciar tres tipos diferentes de *event handlers*.

3.1. Manejadores de eventos en etiquetas HTML

En las primeras versiones de HTML se le asignaron a los elementos unos atributos con los mismos nombres que los eventos, y los cuales pueden detectar estos eventos y reaccionar a ellos. Estos atributos reciben como valor la función que se quiere ejecutar.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <button type="button" onclick="console.log('Hola');">Saludar</button>
</body>
</html>
```

Esta primera forma es una **mala práctica** y hay que evitar usarla porque ensucia el código HTML. Debería de estar separado el código JavaScript del código HTML.

3.2. Manejadores de eventos en el DOM

Los event handlers del DOM nos permiten separar el código JavaScript del código HTML de una forma sencilla. La desventaja de este método es que solo podemos asignar una función a cada evento para un mismo elemento.

Para poder usarlo, tendremos que igualar la *referencia* a la función que se quiere ejecutar (o una función anónima) al `elemento.onevent`. Se le asigna la referencia a la función (sin parentesis) para que no se ejecute la función hasta que se dispare el *event handler*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <button id="btn" type="button">Saludar</button>
</body>
<script>
  var btn = document.getElementById('btn');
  btn.onclick = saludar;

  function saludar() {
    console.log('Hola');
  }
</script>
</html>
```

3.3. Event Listeners

Los **event listeners** se introdujeron como mejora en la especificación del DOM que salió en el año 2000. Es la forma que más se usa para manejar los eventos. La sintaxis es algo distinta a la de los métodos anteriores. Este método soluciona el problema del anterior, que no podía asignarse nada más que una función a un evento para el mismo elemento. La desventaja de este método es que los navegadores antiguos no lo soportan.

En este caso para usarlo tendremos que ejecutar el método `addEventListener()` del elemento, y pasarle como parámetros, el evento que se quiere detectar, y la función que se quiere ejecutar cuando el evento ocurra. En este caso, el evento que se le pasa a la función **no va precedido de *on***.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <button id="btn" type="button">Saludar</button>
</body>
<script>
  var btn = document.getElementById('btn');
  btn.addEventListener('click', saludar);

  function saludar() {
    console.log('Hola');
  }
</script>
</html>
```

Podemos eliminar los *event listeners* asignados a un elemento usando el método `removeEventListener()` del elemento, pasandole los mismos parámetros que cuando se lo hemos asignado.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <button id="btn1" type="button">Quitar event listener del boton 2</button>
  <button id="btn2" type="button">Saludar</button>
</body>
<script>
  function saludar() {
    console.log('Hola');
  }

  var btn2 = document.getElementById('btn2');
  btn2.addEventListener('click', saludar);

  var btn1 = document.getElementById('btn1');
  btn1.click = function() {
    btn2.removeEventListener('click', saludar);
  }
</script>
</html>
```

3.4. Ejercicio 1

- Añadir un script en la etiqueta *head* que asigne un *event handler del DOM* a un div para mostrar un mensaje por consola cuando el ratón se mueva sobre ese elemento

3.5. Ejercicio 2

- Cambiar el color del borde de un input cuando tiene el foco, y dejarlo como estaba cuando pierde el foco

4. Objeto Event

Cuando ocurre un evento podemos acceder a la información del evento a través del objeto **Event** que se crea. Este objeto lo recibimos como parámetro en la función de callback que se ejecuta cuando se detecta dicho evento. Este objeto puede darnos información como sobre que elemento ha ocurrido el evento (propiedad `target`), que tipo de evento se ha disparado (propiedad `type`)...

4.1. Ejercicio 3

- Mostrar en una elemento HTML, todos los códigos de las teclas que se han ido pulsando al escribir sobre un elemento *input*
- Restringir el campo de texto a 200 carácteres
- Mostrar el número de carácteres que quedan por introducir
- Actualizar el número de carácteres que quedan por introducir cada vez que se pulsa una tecla que añade cualquier carácter en el campo

4.2. Cambiando el comportamiento por defecto de los eventos

El objeto **event** tiene algunos métodos que permiten cambiar el comportamiento que tienen por defecto algunos elementos frente a unos eventos y también permite cambiar la forma en que los elementos superiores actuan frente a un evento de un componente que es descendiente de estos.

4.2.1. preventDefault()

Algunos eventos tienen comportamientos por defecto, como hacer *click* en un *link* que nos lleva a otra página. Estos comportamientos se pueden evitar usando el método `preventDefault()`.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <a id="link" href="www.toggl.com">Ir a Toggl</a>
</body>
<script>
  var link = document.getElementById('link');
  link.addEventListener('click', function(e) {
    e.preventDefault();
    console.log('Deberías de haber ido a la página, pero hemos quitado el
comportamiento por defecto');
  });
</script>
</html>
```

4.2.2. stopPropagation()

A veces nos encontramos con que dos elementos de HTML (uno dentro del otro) tienen un *event listener* escuchando al mismo evento. En este caso cuando ocurre ese evento en el elemento que está más abajo en el árbol, se disparará su *event handler* y todos los *event handlers* de los elementos que lo contienen y es muy posible que ocurra un efecto no deseado. Con el método `stopPropagation()` se consigue parar la propagación del evento hacía arriba.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <div id="caja">
    <button type="button" id="btn">Pulsa</button>
  </div>
</body>
<script>
  var caja = document.getElementById('caja');
  var btn = document.getElementById('btn');

  caja.addEventListener('click', function(e) {
    console.log('Has pulsado en el div!!!');
  });

  btn.addEventListener('click', function(e) {
    e.stopPropagation();
    console.log('Has pulsado en el botón!!!');
  });
</script>
</html>
```

4.3. Ejercicio 4

- Crea un botón dentro de una caja
  - Añade un event handler a cada uno para detectar el evento *click*
    - Al pulsar sobre la caja, esta tiene que cambiar su color de fondo por un color aleatorio
    - Al pulsar sobre el botón, este tiene que cambiar el color de fondo por un color aleatorio
    - Al pulsar sobre el botón la caja no tiene que cambiar de color

4.4. Ejercicio 5

- Hacer el juego de piedra, papel o tijera

4.5. Ejercicio 6

- Mover un div por la página usando las flechitas del teclado