

# {JS}

Clase 26

JS

# <índice>

## Prototipos

**¿Qué es un evento?**

---

**¿Cómo capturarlos?**

---

**Eventos mediante HTML**

---

**Eventos mediante Javascript**

---

**Eventos con addEventListener**

---

**El objeto Event**

**¿Qué es un evento?**

# ¿Qué es un evento?

Los eventos son el idioma en el que JavaScript entiende las acciones del usuario. Es decir, son el mecanismo por el que se consigue establecer una comunicación bidireccional y en tiempo real entre la aplicación y los usuarios.

Tal y como su nombre indica, un evento no es más que un suceso que se ha producido en la página, normalmente provocado por la acción del usuario.

## ¿Qué es un evento?

Lo más interesante de este mecanismo es que es asíncrono, por tanto, no hay que esperar por él.

Simplemente se prepara el código para que cuando se produzca el evento el programa lo capture y ejecute la lógica que interese.

Los eventos se asocian a elementos concretos del DOM. En el argot de los programadores se dice que «escuchamos» a ese elemento, o lo que es lo mismo, tenemos un programa preparado para capturar el evento que lanza ese elemento.

**¿Cómo los capturamos?**

# ¿Cómo los capturamos?

Mediante HTML

Mediante Javascript

Mediante addEventListener

# **Mediante HTML**

```
<script>
  function doTask() {
    alert("Hello!");
  }
</script>

<button onClick="doTask()">Saludar</button>
```

```
<script src="tasks.js"></script>
<button onClick="doTask()">Saludar</button>
```

¿problemas?

Ahora aparece un nuevo problema que quizás puede que aún no sea muy evidente. En nuestro <button> estamos haciendo referencia a una función llamada doTask() que, aparentemente, confiaremos en que se encuentra declarada dentro del fichero tasks.js.

Esto podría convertirse en un problema, si posteriormente, o dentro de cierto tiempo, nos encontramos modificando código en el fichero tasks.js y le cambiamos el nombre a la función doTask(), ya que podríamos olvidar que hay una llamada a una función Javascript en uno (o varios) ficheros .html.

# Mediante Javascript

Existe una forma de gestionar eventos Javascript sin necesidad de hacerlo desde nuestros ficheros .html. No obstante, se trata de una «trampa», puesto que seguimos haciéndolo desde HTML, sólo que ese HTML se crea desde Javascript, y nos permite llevárselo a los ficheros .js.

```
<button>Saludar</button>

<script>
const button = document.querySelector("button");
button.onclick = function() {
  alert("Hello!");
}
</script>
```

Observa que en este caso, en lugar de añadir el atributo onClick a nuestro <button>, lo que hacemos es localizarlo mediante querySelector(). Esto podríamos hacerlo mediante una clase, pero en este ejemplo lo hemos hecho directamente mediante el botón, para simplificar.

¿problemas?

Los mismos que antes

es similar a =>

```
<button>Saludar</button>

<script>
const button = document.querySelector("button");
const doTask = () => alert("Hello!");
button.setAttribute("onclick", "doTask()");
</script>
```

A grandes rasgos, se trata de una forma alternativa a gestionar los eventos Javascript desde HTML, pero creando el HTML mediante la API del DOM de Javascript.

**El siguiente método es muuucho mejor**

# **Mediante 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)

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

Más organizado

```
const button = document.querySelector("button");
function action() {
  alert("Hello!");
};
button.addEventListener("click", action);
```

# Con Arrow Functions

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

**Con múltiples listener**

```
<button>Saludar</button>

<style>
  .red { background: red }
</style>

<script>
const button = document.querySelector("button");
const action = () => alert("Hello!");
const toggle = () => button.classList.toggle("red");

button.addEventListener("click", action);          // Hello message
button.addEventListener("click", toggle);          // Add/remove red CSS
</script>
```

# Eliminando EventListener

Se usa para 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.

```
<button>Saludar</button>

<style>
  .red { background: red }
</style>

<script>
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
</script>
```

Si la funcionalidad no la tenemos en una función y hemos creado una función anónima para ello, no podremos usar `removeEventListener` porque serán funciones diferentes

# **El objeto Event**

Para utilizar este objeto, que está asociado a cualquier evento, simplemente hay que indicarlo como parámetro del callback que gestiona el evento. Así, puede reescribirse el listener anterior para hacer uso de él

# Propiedades del objeto Evento

Propiedad	Utilidad
<b>altKey</b>	Devuelve si la tecla [Alt] fue pulsada durante el evento.
<b>button</b>	Devuelve el botón del ratón que activó el evento:
	<ul style="list-style-type: none"> <li>■ 0: botón principal.</li> <li>■ 1: botón central.</li> <li>■ 2: botón secundario.</li> <li>■ 3 y 4: cuarto y quinto botones (si los hubiera).</li> </ul>
<b>charCode</b>	Contiene el valor Unicode de la tecla que se pulsó (evento <b>keypress</b> ).
<b>clientX</b>	Coordenada X del ratón con respecto a la ventana.
<b>clientY</b>	Coordenada Y del ratón con respecto a la ventana.
<b>ctrlKey</b>	Devuelve si la tecla [Ctrl] fue pulsada durante el evento.
<b>pageX</b>	Coordenada X del evento, relativa al documento completo.
<b>pageY</b>	Coordenada Y del evento, relativa al documento completo.
<b>screenX</b>	Coordenada X del evento con respecto a la pantalla.
<b>screenY</b>	Coordenada Y del evento con respecto a la pantalla.
<b>shiftKey</b>	Devuelve si la tecla [Mayús] fue pulsada durante el evento.
<b>target</b>	Referencia al elemento que lanzó el evento.
<b>timeStamp</b>	Devuelve el momento en el que se creó el evento.
<b>type</b>	Nombre del evento.

# **DOMContentLoaded**

Investiga...

# <Despedida>

Email

**bienvenidosaez@gmail.com**

Instagram

**@bienvenidosaez**

Youtube

**youtube.com/bienvenidosaez**