ThePower business school

Eventos JS 🎡🤖

Hasta ahora todo el **código** JavaScript que hemos generado se **ejecutaba** “directamente” al **cargar** la **página**, pero con esto **no** conseguimos **ofrecer al usuario** demasiada **interacción**.

Para cubrir nuestras necesidades de interacción podemos hacer uso de **eventos**, que una vez detectados **lancen las funciones** que les indiquemos.

Evento en html

Un **evento Javascript** es una característica especial que ha sucedido en nuestra página y a la cuál le asociamos una funcionalidad, de modo que se ejecute cada vez que suceda dicho evento. Por ejemplo, el evento **click** se dispara cuando el usuario hace click en un elemento de nuestra página. Imaginemos el siguiente código HTML:

```
<button>Saludar</button>
```

En nuestro navegador nos aparecerá un botón con el texto «**Saludar**». Sin embargo, si lo pulsamos, no realizará ninguna acción ni tendrá funcionamiento. Para solucionar esto, podemos asociarle un evento:

```
<button onClick="alert('Hello!')">Saludar</button>
```

En este ejemplo, cuando el usuario haga click con el ratón en el botón **Saludar**, se disparará el evento **click** en ese elemento HTML (**<button>**). Dicho botón, al tener un atributo **onClick** (*cuando hagas click*), ejecutará el código que tenemos asociado en el

valor del atributo HTML (*en este caso un **alert()***), que no es más que un mensaje emergente con el texto indicado.

Eventos en el DOM

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 llevarlo a los ficheros **.js**.

La idea es la misma que vimos en el punto anterior, sólo que en esta ocasión haremos uso de una propiedad Javascript, a la que le asignaremos la función con el código asociado.

```
<button>Saludar</button>

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

Realmente lo que estamos haciendo es equivalente a añadir un atributo **onclick** en nuestro **<button>**, solo que lo hacemos a través de la API de Javascript. Otra forma similar, donde si se verá más claro, sería la siguiente.

```
<button>Saludar</button>

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

Escuchadores de eventos

Otra manera de tratar los eventos, sería vinculando un **eventListener** (escuchador de eventos). De esta manera preparamos la programática necesaria para que cuando se dispare un evento concreto, se gestione en la función que nosotros hemos preparado para ello.

```
<body>
  <p>Justo debajo encontramos un botón. Prueba a pulsarlo.</p>
  <button id="btn">Púlsame</button>
</body>

let handleClick = function(event) {
   console.log(event);
}
document.getElementById("btn").addEventListener("click", handleClick);
```

Siguiendo las buenas prácticas, más adelante veremos cómo gestionar estos **eventListeners** y procuraremos NO utilizar **onclick** en el HTML (para desacoplar la vista de la capa lógica).