DOM - Eventos

Una de las mayores características de JavaScript es poder reaccionar a eventos, tanto del propio sitio web como del usuario. Existen decenas de eventos en función de los elementos a los que se lo apliquemos, de si tienen animaciones o transiciones, eventos de usuario, de teclado, de ratón…
Lo bueno es que TODOS funcionan igual, en lenguaje humano podríamos traducirlo en “escucha X y cuando lo oigas haz Y”

La sintaxis se divide en tres partes:
element: Elemento que escucha.
event: El evento que escucha.
callback: Qué hará cuando lo escuche, es el nombre de la función a la que llamaremos cuando suceda el evento

```
element.addEventListener('evento', callback)
```

DOM - Eventos

Por ejemplo, si quisiéramos que cuando se haga click en el sitio web se muestre un mensaje por consola diciendo “has hecho click” la sintaxis sería:

```
document.addEventListener('click', ()=>console.log('Has hecho click'))
```

DOM - Eventos

El callback puede ser interno, como en el ejemplo o externo. Ambos hacen exactamente lo mismo.

```
const handleClick = () =>{
  console.log('Has hecho click')
}

document.addEventListener('click', handleClick)
```

La única diferencia es que si es interno, la función no necesita nombre, pero si es externo, necesita un nombre para poder referenciarla.
Es MUY IMPORTANTE que en la función de callback NO pongamos los paréntesis, porque no queremos que se ejecute directamente.

DOM - Objeto Evento

Cada vez que se dispara un evento, automáticamente se envía la información de ese evento como parámetro del callback. Si necesitamos información del evento lo ponemos como parámetro de la función de callback y ya tendremos acceso a él.
Dentro de ese evento tenemos la información de todo lo que podemos leer y/o modificar.

```
document.addEventListener('click', event => {
  console.log(event);
});
```

```
const handleClick = event => {
  console.log(event);
};

document.addEventListener('click', handleClick);
```

```
▶ PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure: 0, …}
```