Material descargable

Javascript: Eventos

egg

¿Qué es un `event listener`?

Un event listener en programación, especialmente en el contexto de JavaScript y la manipulación del Document Object Model (DOM), es una **función que espera y reacciona a un evento en un elemento HTML**. Los eventos pueden ser acciones del usuario como clics, pulsaciones de teclas, movimientos del ratón, o eventos del sistema como la carga de la página. Los event listeners son clave para la programación interactiva en la web.

Algunos aspectos clave:

- ***Escuchar Eventos***: Un event listener "escucha" un tipo específico de evento en un elemento seleccionado.
- ***Reacción a Eventos***: Cuando el evento especificado ocurre, el event listener ejecuta una función de `callback asociada`.
- ***Función de Callback***: La función que se ejecuta en respuesta al evento. Puede ser una función anónima o una función nombrada.
- ***Uso en JavaScript***: En JavaScript, puedes usar el método `addEventListener` para adjuntar un event listener a un elemento. Por ejemplo: `element.addEventListener('click', functionToCall);`. // Función nombrada `element.addEventListener('click', () => {});`. //Función anónima (comúnmente usada) En estas línea estás diciendo: "Cuando el usuario haga click en 'element', ejecuta `functionToCall` o la función anónima"
- ***Remover Event Listeners***: También puedes remover un event listener si ya no es necesario, usando `removeEventListener`.

Los eventos más utilizados

A continuación te contamos cuáles son los eventos más importantes y cuando se ejecutan:

- **DOMContentLoaded**: Se dispara cuando el DOM ha sido completamente cargado y está listo para usar.
- **click**: Se dispara cuando un usuario hace clic en un elemento.
- **change**: Se dispara para elementos de entrada (input, select, textarea) cuando cambia su valor.
- **submit**: Se dispara cuando se envía un formulario.
- **dblclick**: Se activa cuando un usuario hace doble clic en un elemento.

En la función ***callback***, tenemos acceso a una variable que representa el evento disparado.

```
Unset
element.addEventListener("click", (event) => {



})
```

Las propiedades más importante de la variable `event` son:

- **target**: Representa el elemento que disparó el evento. Es muy útil cuando el mismo controlador de eventos se usa para múltiples elementos.
- **preventDefault()**: Es un método, no una propiedad. Llamándolo dentro de un controlador de eventos, puedes prevenir el comportamiento predeterminado del navegador para ese evento.
- **stopPropagation()**: Es otro método importante, que cuando se llama, evita que el evento se propague más allá del elemento actual.

Más información en: Eventos de JS

Función `querySelectorAll`

Para continuar, será esencial familiarizarte con una función clave llamada `querySelectorAll`, que a diferencia de `querySelector`, que solo devuelve el primer elemento que coincide con el selector, `querySelectorAll` devuelve todos los elementos coincidentes.

El tipo de dato que retorna `querySelectorAll` no es un array de JavaScript tradicional, sino un Objeto `NodeList`, aunque es similar a un array en el que puedes acceder a sus elementos por índice.

También tiene una propiedad `length` la cual sirve para iterar sobre él en un bucle. Carece de muchos métodos de los arrays, como `map` o `reduce`, pero sí tiene `forEach`. Por suerte, se puede transformar a un array de JavaScript de la siguiente manera:

```
Unset
const nodeList = document.querySelectorAll('div'); // Obtiene todos los `div` de la página


// Convertirlo a un array

const array = Array.from(nodeList);
```

o bien:

```
Unset
const nodeList = document.querySelectorAll('div'); // Obtiene todos los `div` de la página


// Convertirlo a un array

const array = [...nodeList]; // Uso del operador `spread`, el cual copia los valores de `nodeList` en un array nuevo.
```

Más información en: QuerySelectorAll, NodeList, NodeList - forEach