

DOM

DOM significa Document Object Model (Modelo de Objetos del Documento)

Es la representación en memoria (en forma de árbol de objetos) que el navegador crea cuando carga una página HTML.

Cada etiqueta HTML se convierte en un objeto JS manipulable:

- Un <div> se convierte en un objeto HTMLDivElement.
- Un <p> se convierte en HTMLParagraphElement.
- Y así sucesivamente.

Entonces, **cuando tu HTML se carga**, el navegador lo convierte internamente en algo parecido a esto:

```
<html>
  <head>...</head>
  <body>
    <h1>Hola</h1>
    <p id="mensaje">Bienvenido</p>
    <button>Saludar</button>
  </body>
</html>
```

El navegador lo traduce internamente a un **árbol DOM**:

document

```
└─ html
   └─ head
   └─ body
      └─ h1
      └─ p#mensaje
      └─ button
```

Y ese árbol es lo que tú puedes **recorrer y modificar desde JavaScript**.

Resumiendo, el DOM es la manera que tiene JavaScript de ver y manipular el HTML que ya está cargado en la página.

¿Para qué se usa el DOM en la práctica?

En el ejemplo llamado “uso_basico_DOM” del aula virtual, vamos a ir viendo las formas más comunes de utilizar el DOM.

Eventos:

Un evento NO es simplemente un “clic”, un “submit” o un “keydown”. Un evento es un objeto generado por el navegador cuando “algo ocurre” en la página.

Ejemplos de acciones que pueden generar un evento:

- el usuario hace clic,
- pulsa una tecla,
- mueve el ratón,
- un formulario intenta enviarse,
- una imagen termina de cargar,
- etc.

Cada vez que uno de estos sucesos ocurre, el navegador crea un **objeto evento** con toda la información sobre lo sucedido y lo entrega a la función que tú hayas definido.

Ejemplo de un objeto evento real:

```
{
  "type": "click",
  "target": "<button id='borrar'>",
  "clientX": 133,
  "clientY": 420,
  "timeStamp": 12345
}
```

Ese objeto contiene toda la información útil para trabajar.

Dentro de las funciones que solemos utilizar, destaca `addEventListener`, una función que se ejecuta cuando ocurre un evento dentro de la página: un clic, un envío de formulario, pulsar una tecla, mover el ratón, cargar la página, etc.

Ojo, no es la única forma de ejecutar código cuando ocurre un evento

MÉTODO ANTIGUO → `onclick=` (menos recomendable)

En HTML:

```
<button onclick="saluda()">Hola</button>
```

En JS:

```
function saluda() {
  alert("Hola");
}
```

MÉTODO MODERNO → `addEventListener` (el estándar actual)

En HTML:

`<button id="btn">Hola</button>`

En JS:

`document.getElementById("btn").addEventListener("click", saluda);`

DIFERENCIAS:

onclick (menos recomendado)	addEventListener (recomendado)
Solo permite registrar un evento btn.onclick = fn1; btn.onclick = fn2; El segundo machaca al primero.	Puedes registrar tantos como quieras : btn.addEventListener("click", fn1); btn.addEventListener("click", fn2);
Mezcla HTML y JS	Separa estructura y lógica
Más difícil de mantener	Estándar moderno
No funciona bien con módulos, frameworks o AJAX	Total compatibilidad

A partir de ahora nos centraremos únicamente en addEventListener, que será la función que utilizemos.