**¿Qué es el DOM y para qué se usa?**

El DOM (Document Object Model), en español sería Modelo de Objetos del Documento. Es una interfaz de programación que representa un documento HTML o XML como un árbol de nodos y objetos. Sirve como un puente entre el código JavaScript y la página web, permitiendo que JavaScript acceda, manipule y cambie la estructura, el estilo y el contenido del documento en tiempo real.

**¿Qué representa el DOM en una página web?**

El DOM es una representación estructurada en forma de árbol de un documento web (HTML o XML). Permite que lenguajes de scripting como JavaScript manipulen la estructura, el estilo y el contenido de la página de forma dinámica. Actúa como un puente entre el código y la página, haciendo posible interactuar con ella en tiempo real.

Qué representa el DOM

* **Una estructura de árbol:**

Representa el documento como un árbol jerárquico de nodos, donde cada nodo puede ser un elemento, un texto o un comentario. El nodo raíz es el propio documento, y sus descendientes son los elementos anidados (como <html>, <body>, <div>, etc.).

* **Un modelo de objetos:**

Cada parte del documento (cada etiqueta, por ejemplo) se convierte en un objeto en memoria al que se puede acceder.

* **Una interfaz de programación:**

Proporciona una interfaz (API) que permite a otros lenguajes de programación interactuar con el documento.

Se manipula el DOM con JavaScript seleccionando elementos mediante métodos como getElementById(), querySelector(), y querySelectorAll(), y luego modificando sus propiedades, estilos y contenido. También se pueden añadir o eliminar elementos, y responder a eventos del usuario.

1. Seleccionar elementos

Antes de manipular, es necesario seleccionar los elementos de la página.

* document.getElementById('id'): Selecciona un único elemento por su ID.
* document.querySelector('selector-css'): Selecciona el primer elemento que coincide con un selector CSS (ej. #mi-id o .mi-clase).
* document.querySelectorAll('selector-css'): Selecciona todos los elementos que coinciden con un selector CSS y los devuelve como una lista.
* document.getElementsByClassName('nombre-clase'): Selecciona todos los elementos con una clase específica.
* document.getElementsByTagName('nombre-etiqueta'): Selecciona todos los elementos con un nombre de etiqueta específico (ej. p o div).

2. Modificar contenido y atributos

Una vez seleccionado, puedes cambiar el contenido o los atributos.

* **Texto:** elemento.textContent = 'Nuevo texto';
* **HTML:** elemento.innerHTML = '<h2>Nuevo título</h2>';
* **Atributos:** elemento.setAttribute('src', 'nueva-imagen.jpg');

3. Modificar estilos

Se pueden cambiar los estilos de forma dinámica.

* **Estilos en línea:** elemento.style.color = 'red'; o elemento.style.backgroundColor = 'blue';.
* **Clases CSS:** Es la forma recomendada para modificar estilos.
  + **Añadir una clase:** elemento.classList.add('nombre-clase');.
  + **Eliminar una clase:** elemento.classList.remove('nombre-clase');.
  + **Alternar una clase:** elemento.classList.toggle('nombre-clase');.

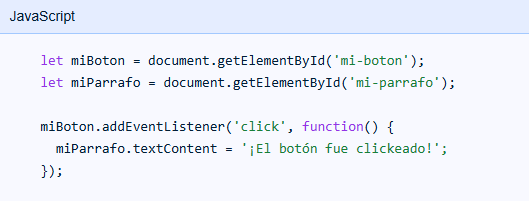
4. Añadir y eliminar elementos

Puedes crear y quitar elementos de la página.

* **Crear un elemento:** let nuevoElemento = document.createElement('div');
* **Añadir un hijo:** elementoPadre.appendChild(nuevoElemento);
* **Eliminar un elemento:** elementoPadre.removeChild(elementoHijo); o elemento.remove();.

5. Manejo de eventos

Se puede hacer que la página reaccione a la interacción del usuario. Añadir un listener de evento.



**Ejemplos prácticos de cómo JS puede cambiar el contenido o estilo de una página usando el DOM.**

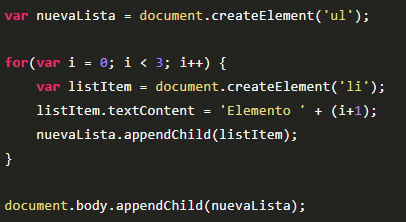
**1er Ejemplo:**

Si tienes un párrafo con id="miParrafo", puedes seleccionarlo así:



**2do Ejemplo:**

Pongamos que deseas crear una nueva lista y añadirla al cuerpo de la página:



**3er Ejemplo:**

La gestión de eventos es clave para la interactividad. Por ejemplo, puedes querer ejecutar una acción cuando alguien hace clic en un botón:

