DOM DE JAVASCRIPT

Carlos Maldonado

5IV7

Introducción

El Document Object Model (DOM) es una de las piezas fundamentales en el desarrollo web moderno. Gracias al DOM, los sitios y aplicaciones web dejan de ser documentos estáticos y se convierten en sistemas dinámicos capaces de responder a la interacción del usuario, actualizar contenido sin recargar la página y comunicarse con servidores en tiempo real.

El DOM actúa como un puente entre el mundo visual de un sitio web —lo que vemos en pantalla— y el lenguaje JavaScript, el cual puede modificar, eliminar, crear o animar elementos dentro de una página.

A lo largo de este trabajo se explicará de manera detallada qué es el DOM, cómo funciona, su estructura, sus nodos, eventos, métodos más utilizados, bibliotecas que lo utilizan y su evolución hacia tecnologías como el Virtual DOM.

¿Qué es el DOM?

El DOM (Document Object Model) es una interfaz de programación que representa un documento HTML o XML como una estructura organizada de objetos. Cada parte de la página web se convierte en un nodo que puede ser manipulado por JavaScript.

En otras palabras, el DOM permite que los lenguajes de programación interactúen con el contenido, estilo y estructura de una página web.

Ejemplo sencillo:

```
document.getElementById("titulo").innerText = "Hola Mundo";
```

Este fragmento reemplaza el texto del elemento con id "titulo".

Historia y origen del DOM

Antes de la existencia del DOM estándar, cada navegador interpretaba HTML a su manera, lo cual generaba incompatibilidades. Netscape y Microsoft compitieron al crear sus propias formas de manipular documentos, lo que originó la necesidad de un estándar universal.

El DOM fue propuesto oficialmente por el W3C como una solución para crear una forma común de representar documentos y permitir su manipulación uniforme desde cualquier navegador.

Con el tiempo, el DOM ha evolucionado a niveles DOM Level 1, 2 y 3, introduciendo más capacidades como manejo de eventos, estilos y nodos avanzados.

Importancia del DOM en el desarrollo web

La importancia del DOM es enorme, ya que sin él los sitios serían totalmente estáticos. Gracias al DOM, los sitios pueden:

- Actualizar información sin recargar página
- Responder a interacciones del usuario
- Crear experiencias dinámicas
- Generar animaciones y transiciones
- Conectar con APIs
- Mostrar contenido según condiciones

Estructura del DOM

El DOM organiza el documento como un árbol jerárquico conocido como árbol DOM. Cada nodo representa una parte del documento.

Estructura básica:

- Document (raíz)
- Elementos (etiquetas HTML)
- Atributos
- Texto
- Comentarios

Funcionamiento del DOM

Cuando un navegador carga una página, primero interpreta el HTML y construye el DOM. Después, JavaScript puede manipularlo.

Acciones comunes:

- Buscar elementos (getElementById, querySelector)
- Modificar texto (innerText, textContent)
- Alterar estilos (style.color = "red")
- Crear nodos (createElement)
- Eliminar nodos (remove())

Esto permite construir experiencias dinámicas y aplicaciones interactivas.

Tipos de nodos en el DOM

- **Document**
- **Element**
- **Text**
- **Attribute**
- **Comment**
- **DocumentFragment** (especial para eficiencia)

Cada nodo tiene propiedades y métodos para interactuar con él.

Métodos y propiedades del DOM

Métodos comunes:

- getElementById()
- querySelector()
- createElement()
- appendChild()
- removeChild()

Estas herramientas permiten manipular la estructura y contenido.

Manejo de eventos

Los eventos permiten que la página responda a acciones.
Ejemplos:

- click
- submit
- mouseover
- keydown
- input

Ejemplo:

```
document.getElementById("btn").addEventListener("click",
() => {
alert("Botón presionado");
});
```

Diferencia entre DOM y BOM

El BOM se refiere al Browser Object Model e incluye objetos como:

- window
- location
- history
- navigator

El DOM solo maneja el documento, el BOM controla propiedades del navegador.

DOM Virtual

Frameworks como React usan Virtual DOM, que crea una copia del DOM en memoria y optimiza la actualización de la interfaz.

Ventajas:

- Mejor rendimiento
- Menos manipulación directa
- Mejor escalabilidad

Bibliotecas y Frameworks

- **jQuery**
- **React** (Virtual DOM)
- **Vue.js** (DOM virtual reactivo)
- **Angular** (motor de plantillas y detección de cambios)

Análisis profundo del DOM y su impacto en la web

El DOM no solo es un recurso técnico; es una tecnología que cambió para siempre la forma de interactuar con los documentos en la web. Antes de su existencia, los sitios eran puramente estáticos y para actualizar contenido era necesario recargar toda la página.

Hoy, gracias al DOM, las páginas son capaces de actualizar porciones específicas en tiempo real, generar contenido nuevo, reaccionar a eventos del usuario y comunicarse con servicios externos.

Relación entre DOM, HTML y CSS

El DOM se construye directamente a partir del HTML, y las propiedades visuales que se modifican afectan el CSS. Es decir:

- **HTML = estructura**
- **CSS = estilos**
- **DOM = representación programable del documento**

JavaScript interactúa con el DOM para modificar HTML y CSS.

Ciclo de Renderizado

Los navegadores siguen un proceso específico al cargar una página:

1. Descarga del HTML
2. Interpretación y construcción del DOM
3. Descarga e interpretación del CSS → creación del CSSOM
4. Unificación DOM + CSSOM → Render Tree
5. Pintado en pantalla

Reflow y Repaint

Modificar el DOM puede generar:

- **Reflow**: recalcular tamaño y posición de elementos
- **Repaint**: volver a dibujar elementos

Cambios constantes al DOM = páginas lentas. Por eso existen técnicas de optimización como DocumentFragment, Virtual DOM y batch updates.

Buenas prácticas al usar DOM

- Minimizar manipulaciones directas
- Usar fragmentos (DocumentFragment)
- Alterar clases en lugar de estilos individuales
- Delegación de eventos en contenedores
- Usar DOM virtual cuando sea necesario

Ejemplo avanzado: Creación dinámica de lista

```
const frutas = ["Manzana", "Banana", "Uva", "Mango"];
const ul = document.createElement("ul");
const fragment = document.createDocumentFragment();
frutas.forEach(fruta => {
const li = document.createElement("li");
li.textContent = fruta;
fragment.appendChild(li);
});

ul.appendChild(fragment);
document.body.appendChild(ul);
```

Seguridad en DOM

Manipulación irresponsable puede causar:

- XSS (Cross Site Scripting)
- Inyección de HTML
- Manipulación maliciosa

Se recomienda usar:

- textContent en vez de innerHTML
- Librerías sanitizadoras

Impacto del DOM en frameworks modernos

Tecnología	Relación con DOM
React	Virtual DOM
Vue	DOM reactivo + Virtual DOM
Angular	Real DOM con sistema de detección
Svelte	Compila a código directo DOM nativo

Caso práctico: Chat dinámico

```
function enviarMensaje() {
const input = document.getElementById("mensaje");
const chat = document.getElementById("chat");
const p = document.createElement("p");
p.textContent = input.value;
chat.appendChild(p);
input.value = "";
}
```

Accesibilidad y DOM

La manipulación del DOM debe realizarse considerando la accesibilidad. La web moderna debe ser usable para todos los usuarios, incluyendo personas con discapacidades visuales, motoras o cognitivas. Para esto es fundamental utilizar buenas prácticas como el uso correcto de etiquetas semánticas (header, nav, main, footer), atributos alt en imágenes, y roles ARIA cuando sea estrictamente necesario.

Buenas prácticas:

- Mantener el orden lógico del DOM para la navegación por teclado
- Utilizar elementos HTML nativos (ej. <button> en vez de un div clickeable)
- Usar atributos ARIA sólo en casos necesarios
- Utilizar aria-live en elementos que actualizan información dinámicamente

Optimización del DOM y rendimiento web

Modificar el DOM puede ser costoso en términos de rendimiento, especialmente en aplicaciones grandes. Algunas técnicas avanzadas para optimización son:

Critical Rendering Path

El navegador construye el DOM y el CSSOM antes de renderizar la página. Minimizar recursos bloqueantes mejora el tiempo de carga.

Carga diferida de scripts

- defer: el script se ejecuta tras el parseo del documento
- async: el script descarga y ejecuta cuando está listo

Minimizar reflows y repaints

- Usar DocumentFragment para insertar muchos nodos
- Cambiar clases en vez de estilos individuales
- Realizar lecturas y escrituras del DOM en bloques

Virtualización

Técnica usada para grandes listas en pantalla —solo se renderiza el contenido visible.

requestAnimationFrame

Usar esta función para animaciones suaves sincronizadas con el navegador.

Depuración y pruebas del DOM

Chrome DevTools

Permite inspeccionar el DOM, escuchar eventos, analizar la performance y detectar memory leaks.

MutationObserver

Sirve para observar cambios en el DOM sin usar ciclos constantes (polling).

Ejemplo:

```
const observer = new MutationObserver(muts => console.log(muts));

observer.observe(document.body, { childList: true, subtree: true });
```

Testing del DOM

- **Jest + DOM Testing Library** para pruebas unitarias
- **Cypress / Playwright** para pruebas de interacción real

Seguridad en el DOM

JavaScript puede ser vulnerable si el DOM se manipula de forma insegura. Riesgo común: **XSS** (Cross-Site Scripting).

Medidas:

- Sanitizar entradas de usuario
- Evitar innerHTML con contenido no confiable
- Usar textContent
- Implementar **Content Security Policy (CSP)**

Server-Side Rendering (SSR) y DOM

Frameworks como Next.js o Nuxt generan HTML desde el servidor. Esto:

Ventajas:

- Mejor SEO
- Menor tiempo al primer renderizado

Desventaja:

- Requiere “hidratación” del DOM en el cliente

BIBLIOGRAFÍA:

- Fundación Mozilla. (2024). *Introducción al DOM*. MDN Web Docs. https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction
- Fundación Mozilla. (2024). *DOM: Document Object Model*. MDN Web Docs. https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model
- Fundación Mozilla. (2024). *Árbol DOM y nodos*. MDN Web Docs. https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/DOM_tree
- Fundación Mozilla. (2024). *Manipulación del DOM con JavaScript*. MDN Web Docs. https://developer.mozilla.org/es/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents
- W3C. (2024). *Modelo de Objetos del Documento (DOM)*. World Wide Web Consortium. https://www.w3.org/DOM/
- EcuRed. (2023). *DOM (Modelo de Objetos del Documento)*. https://www.ecured.cu/DOM
- Programando.la. (2023). *¿Qué es el DOM y cómo funciona en JavaScript?* https://programando.la/que-es-el-dom-javascript/
- Código Facilito. (2022). *Guía completa DOM en JavaScript*. https://codigofacilito.com/articulos/dom-javascript-guia
- Fazt Web. (2022). *Tutorial DOM en JavaScript para principiantes*. https://faztweb.com/contenido/dom-javascript
- EDteam. (2022). *Curso de JavaScript: Trabajando con el DOM*. https://ed.team/cursos/javascript
- Domestika. (2022). *El DOM explicado para desarrolladores web*. https://www.domestika.org/es/blog
- Santander Open Academy. (2023). *Introducción al DOM en JavaScript*. https://www.santanderopenacademy.com/es/blog/dom-javascript