**DOM (Document Object Model)**

Se trata una interfaz de programación para documentos HTML y XML. Proporciona una representación estructurada del documento en memoria y permite a los lenguajes de programación interactuar con la estructura, el estilo y el contenido de los documentos web.

*Conceptos*

* **Representación del documento:** El DOM representa un documento como un árbol jerárquico de nodos. Cada nodo en el árbol representa una parte del documento, como un elemento HTML, un atributo o un texto.
* **Interfaz de programación:** El DOM proporciona una interfaz que permite a los lenguajes de programación (principalmente JavaScript en el contexto de la web) manipular el contenido y la estructura del documento. Esto incluye la capacidad de modificar el contenido de los elementos, cambiar atributos, agregar o eliminar nodos, y responder a eventos del usuario.

*Estructura del DOM*

* **Árbol de nodos:** El documento HTML se transforma en un árbol de nodos donde el nodo raíz es el objeto **document**. Cada nodo puede tener nodos hijos, y cada uno de estos nodos puede tener sus propios nodos hijos, formando una estructura jerárquica.
* **Elemento Nodo:** Representa una etiqueta HTML. Ejemplo: **<div>**
* **Texto Nodo:** Representa el contenido textual dentro de un elemento. Ejemplo: "Hola, mundo!" dentro de **<p>Hola, mundo!</p>.**
* **Atributo Nodo:** Representa un atributo de un elemento HTML. Ejemplo: **class="mi-clase"** en <div class="mi-clase">.
* **Jerarquía del DOM:** La jerarquía del DOM refleja la estructura del documento HTML. El <html> es el nodo raíz, seguido por <head> y <body>, y dentro de <body> pueden estar otros elementos como <div>, <header>, <footer>, etc.

*Manipulación del DOM*

* **Acceso a Elementos:** Con JavaScript, podemos acceder a elementos del DOM utilizando métodos como **getElementById()**, **getElementsByClassName()**, **querySelector()**, etc.
* **Modificación de Elementos:** podemos cambiar el contenido, atributos y estilo de los elementos.
* **Creación y eliminación de elementos:** podemos crear nuevos elementos y agregarlos al DOM, o eliminar elementos existentes.

*DOM y Eventos*

* **Eventos del Usuario:** El DOM permite agregar manejadores de eventos para responder a acciones del usuario, como clics, desplazamientos o cambios en formularios.
* **Eventos y Propagación:** Los eventos en el DOM pueden ser capturados y manejados en diferentes fases: captura, objetivo y burbuja. La propagación de eventos permite gestionar cómo los eventos se manejan en diferentes niveles de la jerarquía del DOM.

**DOM Virtual**

Virtual DOM (en el contexto de bibliotecas como React) es una representación en memoria del DOM real. Permite a las bibliotecas y frameworks realizar cambios en el DOM de manera eficiente, actualizando solo las partes necesarias en lugar de manipular directamente el DOM real en cada cambio. Esto mejora el rendimiento de las aplicaciones al reducir la cantidad de operaciones directas en el DOM.