DOM

¿Qué es el DOM?

DOM significa ***Document Object Model***, en español sería Modelo de Objetos del Documento.

Es una interfaz de programación que nos permite crear, cambiar, o remover elementos del documento.

También podemos agregar eventos a esos elementos para hacer más dinámica nuestra página.

El DOM es una representación estructurada de un documento HTML (o XML) que permite a los programadores acceder, manipular y modificar los elementos de una página web de manera dinámica mediante JavaScript.

Imagina que una página web es como una casa con muchas habitaciones. Cada habitación es un elemento en la página web, como un título, un botón o un párrafo. El DOM es como un plano o mapa de la casa que te permite interactuar con cada habitación de forma individual.

Aquí hay algunos conceptos clave sobre el DOM:

1. Estructura de árbol: El DOM organiza los elementos de una página web en una estructura de árbol jerárquica. El elemento raíz es el documento HTML, y a partir de ahí, los elementos se conectan formando ramas y subramas, como las habitaciones de la casa. Cada elemento se convierte en un nodo en el árbol DOM.

2. Acceso y manipulación: Mediante JavaScript, puedes acceder a los elementos del DOM utilizando métodos y propiedades específicos. Puedes buscar elementos por su etiqueta, clase, ID o cualquier otro atributo. Una vez que accedes a un elemento, puedes cambiar su contenido, estilos, atributos y más.

3. Eventos: El DOM también te permite trabajar con eventos, como hacer clic en un botón o mover el mouse sobre un elemento. Puedes agregar controladores de eventos para responder a estas acciones y realizar acciones específicas en consecuencia.

4. Creación y eliminación de elementos: Puedes crear nuevos elementos HTML y agregarlos al DOM, así como eliminar elementos existentes. Esto es útil cuando deseas agregar contenido dinámicamente o manipular la estructura de la página.

5. Actualización en tiempo real: Una vez que modificas el DOM mediante JavaScript, los cambios se reflejan automáticamente en la página web. Esto te permite crear experiencias interactivas y dinámicas para los usuarios.

6. Navegación y búsqueda: El DOM te permite navegar a través de los elementos y buscar elementos específicos dentro de la estructura de la página. Puedes acceder al padre, hijo o hermano de un elemento, lo que facilita la manipulación de elementos

relacionados. También puedes realizar búsquedas más avanzadas utilizando selectores CSS, lo que te permite seleccionar elementos con mayor precisión.

7. Manipulación de estilos: Además de modificar el contenido de los elementos, puedes cambiar los estilos de un elemento a través del DOM. Puedes modificar propiedades como el color, tamaño, posición y visibilidad de los elementos. Esto te permite crear efectos visuales y personalizar la apariencia de tu página web dinámicamente.

8. Interacción con formularios: El DOM también te permite interactuar con formularios en una página web. Puedes acceder y modificar los valores de los campos de entrada, validar datos antes de enviar un formulario y realizar acciones específicas cuando se envía un formulario.

9. Manipulación de atributos: A través del DOM, puedes acceder y modificar los atributos de los elementos HTML. Puedes agregar, eliminar o cambiar los valores de los atributos, lo que te permite personalizar el comportamiento y las características de los elementos de la página.

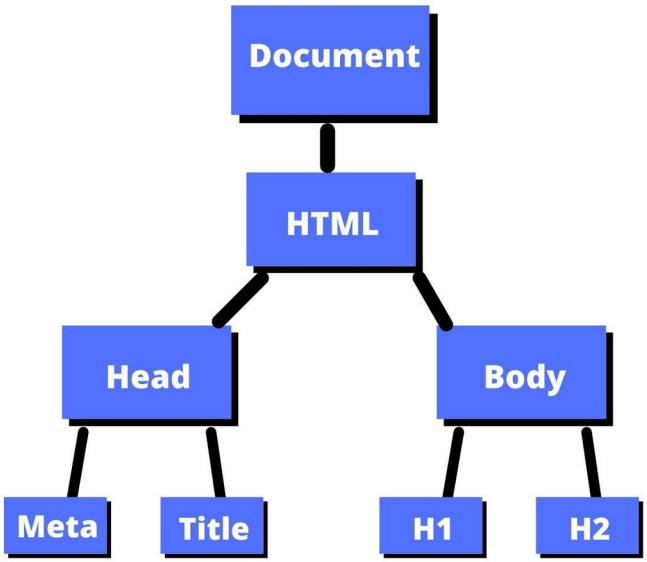
10. Renderizado en múltiples dispositivos: Una gran ventaja del DOM es que es independiente de la plataforma y del navegador. Esto significa que puedes utilizar los mismos métodos y propiedades del DOM en diferentes navegadores web, lo que facilita el desarrollo multiplataforma y garantiza una experiencia consistente en diferentes dispositivos.

Representación en forma de árbol

En resumen, el DOM es una forma de representar y manipular la estructura y contenido de una página web mediante JavaScript. Te permite acceder y modificar elementos, responder a eventos y crear contenido de forma dinámica. Es una herramienta fundamental para el desarrollo web interactivo.

El DOM visualiza el documento de HTML como un árbol de **tres** nodos.

- Un nodo representa un documento de HTML.
- Nuestro documento se llama nodo raíz y contiene un nodo hijo el cual es el elemento <html>.
- El elemento <html> contiene dos hijos los cuales son los elementos <head> y <body>.
- Ambos elementos <head> y <body> tienen hijos propios.



Podemos acceder a estos elementos en el documento y hacer cambios a ellos usando JavaScript.

Como seleccionar Elementos en el Documento

Hay diferentes métodos para seleccionar un elemento en el documento de HTML.

Nos enfocaremos en tres de esos métodos:

- **getElementById()**
- **querySelector()**
- **querySelectorAll()**

a) getElementById()

En HTML, los ids se utilizan como identificadores únicos para los elementos HTML. Esto significa que no podemos tener el mismo nombre de id para dos elementos diferentes.

Esto sería incorrecto:

```
<p id="para">Este es mi primer parágrafo.</p>
<p id="para">Este es mi segundo parágrafo.</p>
```

Tendrías que asegurarte de que esos id sean únicos como los siguientes:

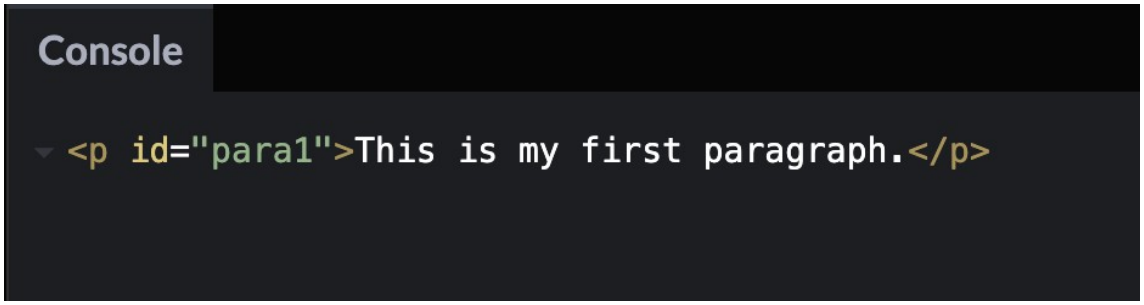
```
<p id="para1">Este es mi primer parágrafo.</p>
<p id="para2">Este es mi segundo parágrafo.</p>
```

En JavaScript, podemos tomar una etiqueta de HTML haciendo referencia al nombre del id.

```
document.getElementById("nombre de id va aquí")
```

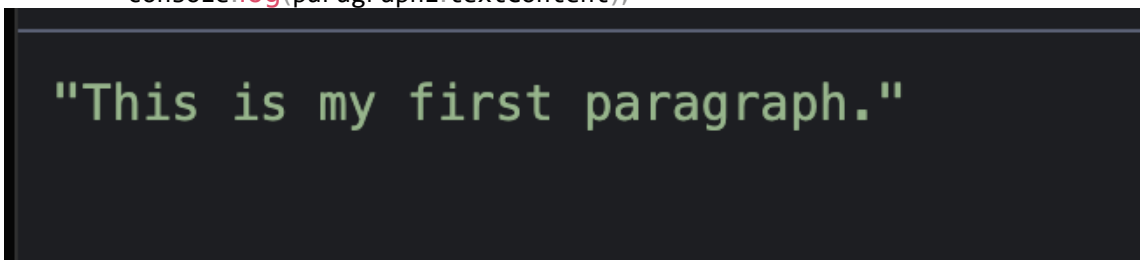
Este código le dice a la computadora que obtenga el elemento <p> que tiene el id de para1 y que imprima el elemento en la consola.

```
const paragraph1 = document.getElementById("para1");
console.log(paragraph1);
```



Si deseamos leer solamente el contenido del párrafo, entonces podemos usar la propiedad textContent dentro del console.log().

```
const paragraph1 = document.getElementById("para1");
console.log(paragraph1.textContent);
```



b) querySelector()

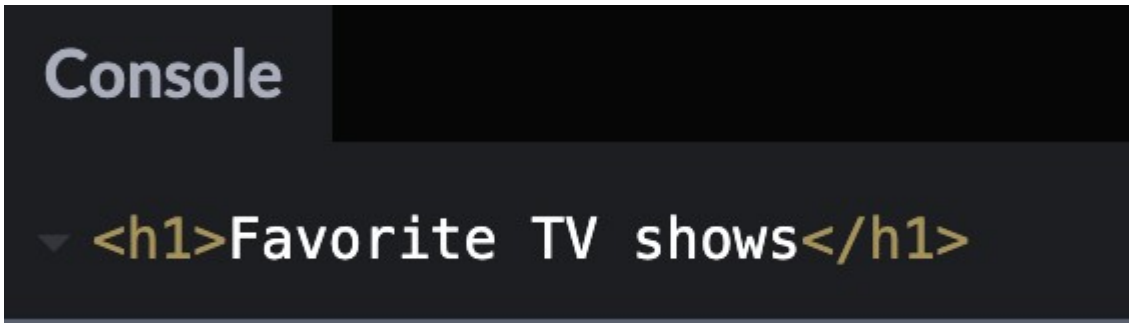
Puedes usar este método para encontrar elementos con uno o más selectores de CSS.

Ejemplo de programas favoritos:

```
<h1>Favorite TV shows</h1>
<ul class="list">
  <li>Golden Girls</li>
  <li>Archer</li>
  <li>Rick and Morty</li>
  <li>The Crown</li>
</ul>
```

Si deseamos encontrar e imprimir el elemento h1 , entonces podemos usar ese nombre de etiqueta dentro del querySelector().

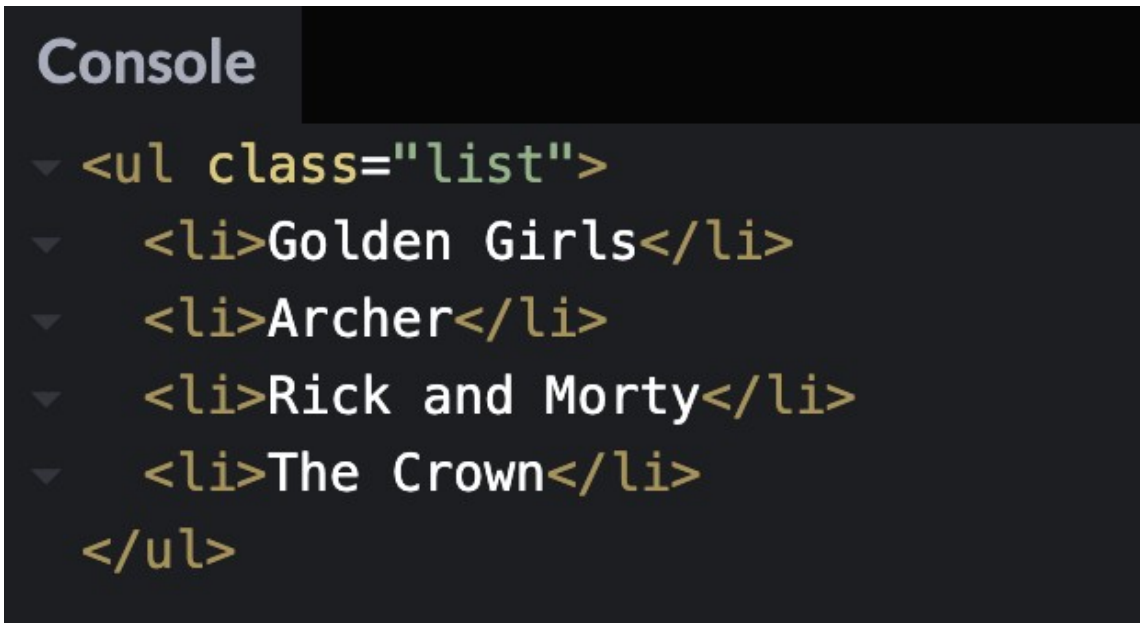
```
const h1Element = document.querySelector("h1");
console.log(h1Element);
```



Si quisiera apuntar a class="list" para imprimir la lista no ordenada en la consola, entonces usaría .list dentro del querySelector().

El . antes de list le dice a la computadora que apunte al nombre de la Clase. Si deseas apuntar a un id entonces te recomiendo usar el símbolo # antes del nombre.

```
const list = document.querySelector(".list");
console.log(list);
```



c) querySelectorAll()

Este método encuentra todos los elementos que coinciden con el selector de CSS y devuelve una lista de todos esos nodos.

Si quisiera encontrar todos los elementos <li> en nuestro ejemplo, podría utilizar el combinador de hijos > para encontrar a todos los elementos hijos de <ul>.

```
const listItems = document.querySelectorAll("ul > li");
console.log(listItems);
```

Si quisiéramos imprimir los elementos reales <li> con los shows de tv, podemos usar el ciclo forEach() para iterar sobre la NodeList e imprimir cada uno de los elementos.

```
const listaDeElementos = document.querySelectorAll("ul > li");
listaDeElementos.forEach((item) => {
  console.log(item);
});
```