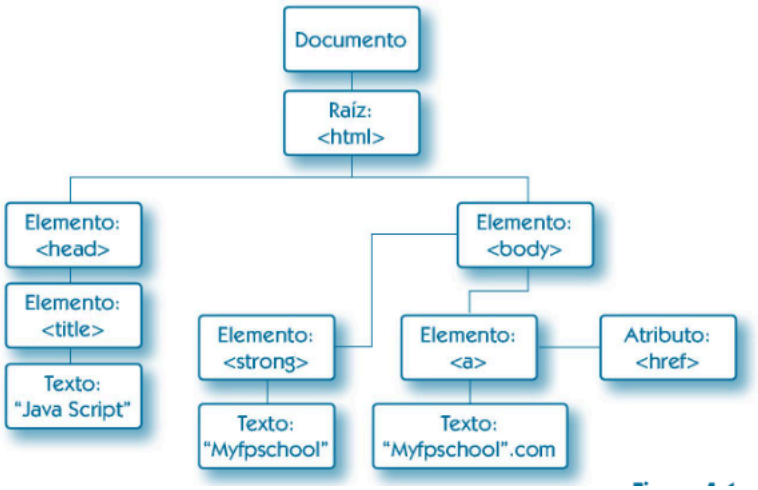
Tema 2. DOM (Document Object Model)

El DOM (Document Object Model) es una representación jerárquica de un documento HTML/XML que proporciona una forma estructurada de interactuar con la página web mediante programación.



1. Estructura Jerárquica

El DOM organiza los elementos HTML en una estructura de árbol, donde cada elemento es un "nodo" y tiene padres, hijos y hermanos.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo DOM</title>
</head>
<body>
  <div id="contenedor">
    <h1>Título Principal</h1>
    <p>Este es un párrafo.</p>
    <ul>
      <li>Elemento 1</li>
      <li>Elemento 2</li>
      <li>Elemento 3</li>
```

```
    </ul>
  </div>
</body>
</html>
```

```
- document
 - html
  - head
   - title
    - "Ejemplo DOM"
  - body
   - div (id="contenedor")
    - h1
     - "Título Principal"
    - p
     - "Este es un párrafo."
    - ul
     - li
      - "Elemento 1"
     - li
      - "Elemento 2"
     - li
      - "Elemento 3"
```

- **Document Object:** El nodo raíz del DOM es el objeto **document**, que representa toda la página web. Es el punto de entrada para interactuar con el DOM. Es el elemento fundamental del DOM (Document Object Model). Representa todo el contenido de un documento HTML o XML cargado en el navegador web.
- **Interacción con la Página**: El objeto document proporciona una interfaz para interactuar y manipular todos los elementos y contenido de la página web. Esto incluye la capacidad de acceder a elementos HTML, modificar su contenido, agregar o eliminar elementos, cambiar estilos, y mucho más.
- **Representación de la Página**: El objeto document ofrece una representación estructurada de la página web. Puedes acceder a cualquier parte de la página a través de este objeto y realizar acciones sobre ella.

2. Nodos y Elementos. Acceso a elementos

Puedes acceder a los elementos del DOM mediante métodos:

- **getElementById()**:
  - Selecciona un elemento por su atributo id.
  - Devuelve el primer elemento con el id especificado.

    const miElemento = document.getElementById('miId');

EJERCICIO A: Crea una página web sencilla que contenga un título y un párrafo. Utilizando JavaScript, cambia el texto del título a "¡Hola Universo!", cambia el color del texto a azul y añade una clase llamada "nuevaClase".

- **getElementsByClassName()**:
  - Selecciona elementos por su atributo class.
  - Devuelve una lista de todos los elementos que tienen la clase especificada.

const elementos = document.getElementsByClassName('miClase');

EJERCICIO B: Crea una página web con una lista de elementos <li>. Asigna la misma clase a todos los elementos de la lista. Utilizando JavaScript, cambia el color de fondo DE FONDO de todos los elementos con esa clase y el color del texto a rojo

- **getElementsByTagName()**:
  - Selecciona elementos por su nombre de etiqueta.
  - Retorna una lista de todos los elementos con la etiqueta especificada.

    const elementos = document.getElementsByTagName('p');

EJERCICIO C. Crea una página web que contenga una lista de películas. La lista debe estar dentro de un elemento <ul> con el id listaPeliculas. Utilizando JavaScript, al presionar un botón se debe cambiar el color del texto de todos los elementos de la lista a verde.

- **Acceso por Índice**. Si tienes una NodeList (como la que se obtiene con getElementsByClassName o querySelectorAll), se puede acceder a elementos individuales por su índice.

  const miElemento = elementos[0]; // Accede al primer elemento

Enunciado D: Crea una página web que contenga una lista de elementos <li> con una clase común, por ejemplo, "elemento". Utilizando JavaScript, recorre todos los elementos de la lista y realiza una operación con cada uno de ellos, como imprimir su contenido en la consola del navegador. Puedes utilizar un bucle for para este propósito.

2.1. querySelector

querySelector devuelve el primer elemento que encuentre y que concuerde con el selector que se introduce como parámetro. Solamente devuelve el primer elemento. Para seleccionar todos los elementos se utiliza querySelectorAll().

- **Seleccionar por Etiqueta:** Este código selecciona todos los elementos <p> en el documento.
  const parrafos = document.querySelectorAll('p');

- **Seleccionar por Clase: Esto selecciona todos los elementos que tienen la clase "mi-clase".**
  const elementosClase = document.querySelectorAll('.mi-clase');

- **Seleccionar por ID:** Aquí, seleccionamos el elemento con el ID "mi-id".
  const elementoId = document.querySelector('#mi-id');

- **Combinar Selectores:** Esto selecciona la lista desordenada (<ul>) con el ID "mi-lista".
  const elemento = document.querySelector('ul#mi-lista');

- **Seleccionar por Atributos:** Esto selecciona el primer enlace (<a>) con el atributo href que apunta a "https://www.ejemplo.com"
  const enlace = document.querySelector('a[href="https://www.ejemplo.com"]');

- **Seleccionar el Primer Elemento que Coincide:** Esto selecciona el primer párrafo en el documento.
  const primerParrafo = document.querySelector('p');

- **Seleccionar un Elemento Dentro de Otro Elemento:** Esto selecciona la lista (<ul>) que está dentro de un con el ID "mi-div"

  const listaDentroDiv = document.querySelector('div#mi-div ul');

- **querySelectorAll()**: devuelve una NodeList que contiene todos los elementos que coinciden con el selector.

  const elementos = document.querySelectorAll('.miClase');

Las pseudoclases son selectores que permiten seleccionar elementos basados en su estado o posición en relación con el documento. Aquí tienes algunos ejemplos de pseudoclases con `querySelectorAll`:

- **Seleccionar el primer elemento <p> dentro de un contenedor:**

  const primerParrafo = document.querySelectorAll('div p:first-child');

- **Seleccionar el último elemento <li> dentro de una lista (<ul>):**

  const ultimoItem = document.querySelectorAll('ul li:last-child');

- **Seleccionar todos los enlaces (<a>) que tienen el estado :hover (cuando el ratón está sobre ellos):**

  const enlacesHover = document.querySelectorAll('a:hover');

- **Seleccionar todos los elementos <input> que están marcados (:checked):**

  const inputsMarcados = document.querySelectorAll('input:checked');

- **Seleccionar todos los elementos <input> que están vacíos (:empty):**

  const inputsVacios = document.querySelectorAll('input:empty');

Para seleccionar los últimos elementos con querySelectorAll, se puede usar la pseudo-clase :nth-last-child(). Esta pseudo-clase selecciona elementos que son el enésimo hijo desde el final de su padre.

- **// Selecciona los últimos dos elementos con la clase "mi-clase"**
- **const elementos = document.querySelectorAll('.mi-clase:nth-last-child(-n+2)');**

elementos es una NodeList que contiene los últimos dos elementos que tienen la clase “mi-clase”. La numeración comienza desde 1, por lo que -n+2 selecciona el último y el penúltimo elemento. Si quisieras seleccionar solo el último elemento, usarías -n+1.

Ejercicio con querySelectorAll E: "Crear una página web que incluya una lista de elementos `<div>`, cada uno con una clase específica (`class`). Utiliza el método `querySelectorAll` para seleccionar y mostrar en la consola los siguientes elementos:

- **Todos los elementos con la clase "item".**
- **Los primeros tres elementos con la clase "destacado".**
- **Los últimos dos elementos con la clase "oculto".**

Ejercicio F: "Crear una página web que incluya un título <h1>, un contenedor con un identificador único (id), dos párrafos <p>, uno de los cuales tiene una clase específica (class), y una lista desordenada <ul> con tres elementos de lista <li>. Utiliza el método querySelector para seleccionar y mostrar en la consola los siguientes elementos:

- **El primer elemento `<h1>`.**
- **El elemento con el identificador único “contenedor".**
- **El primer párrafo con la clase “destacado".**
- **El segundo elemento de lista dentro de la lista desordenada.**