4. Traversing (Recorrido) del DOM

Para utilizar el DOM, es importante saber cómo navegar entre los diferentes nodos.

4.1. parentNode

parentNode: La propiedad parentNode te permite acceder al nodo padre de un elemento**.**

```
const elementoHijo = document.getElementById('hijo');
const padre = elementoHijo.parentNode;
```

4.2. childNodes y children

childNodes y children: permiten acceder a los nodos hijos de un elemento padre, pero hay diferencias clave en su funcionamiento y en el tipo de nodos que devuelven.

1. childNodes:

- **childNodes**: es una propiedad de un elemento en el DOM que devuelve una lista de todos los nodos hijos, incluyendo nodos de texto, nodos de comentario y otros elementos HTML.
- Los nodos dentro de `childNodes` se numeran en función de su posición en el DOM, comenzando desde 0.

```
const padre = document.getElementById("contenedor");
const nodosHijos = padre.childNodes;
```

2. children:

- **children**: es una propiedad similar a `childNodes`, pero devuelve una lista de solo los nodos hijos que son elementos HTML. Es decir, excluye los nodos de texto y otros tipos de nodos no elementales.
- Los nodos dentro de `children` se numeran en función de su posición en el DOM, comenzando desde 0.

```
const padre = document.getElementById("contenedor");
const elementosHijos = padre.children;
```

EJERCICIO A: Crea una página web que contenga un div con el ID "contenedor". Dentro del div, agrega una lista de elementos que incluye dos párrafos y dos elementos span. También, coloca un botón en la página con el texto "Modificar Contenedor". Al pulsar en el botón "Modificar Contenedor", crea una función que realice lo siguiente:

- Utiliza la propiedad childNodes para acceder a todos los nodos hijos del div con el ID "contenedor" y muestra esta lista en consola/alert/en el sitio web.
- Utiliza la propiedad children para acceder a todos los elementos hijos del div y muestra esta lista en la consola/alert/en el sitio web.
- Modifica el contenido del primer y segundo párrafo dentro del div, el primero utilizando childNodes y el segundo utilizando children.

4.3. firstChild y lastChild

firstChild y lastChild: son propiedades que se utilizan para acceder al primer y al último nodo hijo de un elemento padre, respectivamente. Estas propiedades permiten la navegación a través de los nodos hijos de un elemento en el árbol DOM.

1. firstChild:

- **firstChild** es una propiedad que se utiliza para acceder al primer nodo hijo de un elemento padre en el árbol DOM.
- El nodo devuelto por `firstChild` puede ser de cualquier tipo, como un elemento HTML, un nodo de texto o un nodo de comentario.

```
const elementoPadre = document.getElementById("contenedor");
const primerHijo = elementoPadre.firstChild;
```

2. lastChild:

- lastChild es una propiedad que se utiliza para acceder al último nodo hijo de un elemento padre en el árbol DOM.
- Al igual que con `firstChild`, el nodo devuelto por `lastChild` puede ser de cualquier tipo, incluyendo elementos HTML, nodos de texto, o nodos de comentario.

```
const elementoPadre = document.getElementById("contenedor");
const ultimoHijo = elementoPadre.lastChild;
```

Nota: debes tener en cuenta que pueden devolver nodos de texto no deseados si hay espacios en blanco o saltos de línea entre los elementos. Para acceder específicamente a elementos HTML como hijos directos, puedes utilizar `children` en lugar de `firstChild` y `lastChild`.

EJERCICIO B: Crea una página web que contenga un div con el ID "contenedor". Dentro del div, agrega una lista de elementos que incluye dos párrafos y dos elementos span. Además, incluye un botón con el texto "Resaltar Primer y Último Párrafo". Cuando el usuario haga clic en el botón, crea una función que realice lo siguiente:

- Al hacer clic en el botón, los primeros y últimos párrafos del contenedor se tienen que resaltar con un fondo amarillo.

4.4. nextSibling y previousSibling

nextSibling y previousSibling: permiten acceder a los nodos hermanos adyacentes de un nodo específico en el árbol DOM. Estas propiedades son útiles para la navegación a través de los nodos del mismo nivel de jerarquía.

1. nextSibling:

- nextSibling es una propiedad que se utiliza para acceder al siguiente nodo hermano (siguiente hermano adjacente) en el árbol DOM.
- El nodo devuelto por `nextSibling` puede ser de cualquier tipo, como un elemento HTML, un nodo de texto, un nodo de comentario o incluso un nodo de espacio en blanco (espacio o salto de línea).
- `nextSibling` recorre el árbol DOM en el mismo nivel de jerarquía, y si no hay más nodos hermanos a la derecha del nodo actual, devolverá `null`.

```
const nodoActual = document.getElementById("miNodo");
const siguienteHermano = nodoActual.nextSibling;
```

2. previousSibling:

- previousSibling es una propiedad que se utiliza para acceder al nodo hermano anterior (hermano adjacente anterior) en el árbol DOM.
- El nodo devuelto por `previousSibling` también puede ser de cualquier tipo, como elementos HTML, nodos de texto, nodos de comentario o nodos de espacio en blanco.
- `previousSibling` recorre el árbol DOM en el mismo nivel de jerarquía, y si no hay más nodos hermanos a la izquierda del nodo actual, devolverá `null`.

```
const nodoActual = document.getElementById("miNodo");
const hermanoAnterior = nodoActual.previousSibling;
```

Es importante tener en cuenta que tanto `nextSibling` como `previousSibling` pueden devolver nodos de texto que representen espacios en blanco o saltos de línea si existen entre los elementos. Por lo tanto, es posible que debas realizar comprobaciones adicionales para asegurarte de que estás obteniendo el tipo de nodo que deseas.

EJERCICIO C: Crea una página web que contenga un `div` con el ID "contenedor". Dentro del div, agrega una lista de elementos que incluye dos párrafos y dos elementos `span`. También, incluye dos botones con los textos "Resaltar Siguientes" y "Resaltar Anteriores".

Cuando el usuario hace clic en el botón "Resaltar Siguientes", crea una función que realice lo siguiente:

- Utiliza `nextSibling` para acceder al nodo hermano siguiente del primer párrafo dentro del div.

- Si el nodo hermano siguiente es un elemento span, haz que se resalte.

Cuando el usuario hace clic en el botón "Resaltar Anteriores", crea otra función que realice lo siguiente:

- Utiliza previousSibling para acceder al nodo hermano anterior al segundo párrafo dentro del div.
- Si el nodo hermano anterior es un elemento span, agrega la clase de estilo "resaltado" a ese elemento.

4.5. nextElementSibling y previousElementSibling

`nextElementSibling` y `previousElementSibling` permiten acceder a los elementos hermanos (nodos de elementos) adyacentes de un elemento específico en el árbol DOM. A diferencia de `nextSibling` y `previousSibling`, estas propiedades se centran en elementos HTML.

```
const primerElemento = document.getElementById('primerElemento');
const siguienteElemento = primerElemento.nextElementSibling;
const elementoAnterior = primerElemento.previousElementSibling;
```

1. nextElementSibling:

- `nextElementSibling` es una propiedad que se utiliza para acceder al siguiente elemento hermano (siguiente hermano adjacente) en el árbol DOM.

- La propiedad `nextElementSibling` devuelve el siguiente elemento HTML hermano y omite nodos de texto, nodos de comentario u otros tipos de nodos no elementales.

- Si no hay más elementos hermanos a la derecha del elemento actual, `nextElementSibling` devolverá `null`.

```
const elementoActual = document.getElementById("miElemento");
const siguienteElementoHermano = elementoActual.nextElementSibling;
```

2. previousElementSibling:

- `previousElementSibling` es una propiedad que se utiliza para acceder al elemento hermano anterior (hermano adjacente anterior) en el árbol DOM.

- La propiedad `previousElementSibling` devuelve el elemento HTML hermano anterior y omite otros tipos de nodos.

- Si no hay más elementos hermanos a la izquierda del elemento actual, `previousElementSibling` devolverá `null`.

```
const elementoActual = document.getElementById("miElemento");
const elementoHermanoAnterior = elementoActual.previousElementSibling;
```

La diferencia clave entre `nextElementSibling` y `previousElementSibling` y `nextSibling` y `previousSibling` es que los primeros se centran exclusivamente en elementos HTML, mientras que los segundos pueden devolver cualquier tipo de nodo. Esto hace que `nextElementSibling` y `previousElementSibling` sean más útiles cuando se necesita navegar a través de elementos HTML específicamente en el mismo nivel de jerarquía.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de Diferencia entre nextElementSibling y nextSibling</title>
</head>
<body>
  <div id="contenedor">
    <p>Primer Párrafo</p>
    <span>Un span</span>
    <!-- Este es un comentario -->
    <p>Segundo Párrafo</p>
    <span>Otro span</span>
  </div>

  <script>
    const elementoActual = document.querySelector("p"); // Seleccionamos el primer párrafo
    const siguienteElementoHermano = elementoActual.nextElementSibling; // Usando nextElementSibling
    const siguienteNodoHermano = elementoActual.nextSibling; // Usando nextSibling

    console.log("nextElementSibling: ", siguienteElementoHermano); // Debería mostrar el segundo párrafo (un elemento)
    console.log("nextSibling: ", siguienteNodoHermano); // Debería mostrar el segundo párrafo (puede ser cualquier tipo de nodo)
  </script>
</body>
</html>
```