DOM API

Consultar elementos

Método DOM API	**Seleccionará El/Los....**	**Regresará.....**	**Iteramos con:**
document.getElementById('id')	Elemento que tenga el 'id'	Objeto (Elemento)	No aplica
document.querySelector('selector')	Primer Elemento que cumpla con 'selector'	Objeto (Elemento)	No aplica
document.getElementsByTagName('tag')	Elementos que tengan la etiqueta 'tag'	Live HTML Collection	for() For(of)
document.getElementsByClassName('clase')	Elementos que tengan la clase 'clase'	Live HTML Collection	for() For(of)
document.getElementsByName('name')	Elementos que tengan el atributo 'name'	Live Node List	for() forEach()
document.querySelectorAll('selector')	Elementos que cumplan con 'selector'	Static Node List	for() forEach()

> **Notas Importantes**:
> **getElementsBy..** : Los métodos GetElementsBy.. regresan una colección viva (Node List/Html Collection)
> **Static:** Cambios en los elementos del DOM, No se cambiarán en la HTML Collection o Node List
> **Live:** Cambios en los elementos del DOM, Se cambiarán automáticamente en la HTML Collection o Node List

**Propiedades de Elementos que regresan colecciones:

Propiedad	**Regresará**
elemento.childNodes	Live Node List
elemento.children	Live Html Collection

DOM API (HTML Collection y Node List)

HTML Collection

- Es una colección de nodos del DOM que contiene solo nodos de Elemento (las etiquetas del documento)
- Es Live, esto es: Si cambian los elementos del DOM, se cambiarán también de la HTML Collection.
- Es un array-like, no tiene los métodos de los arreglos, pero tiene propiedad length y accesamos a los elementos con corchetes [n]
- Contiene solo dos métodos:
  - `item(n)` Regresa el elemento en el indice n que se le especifique
  - `namedItem(name)` Regresa el primer elemento que tenga el atributo id o name que se le especifique o null si no hay nada.

Node List

- Es una colección de nodos del DOM que contiene cualquier tipo de nodo: de Elemento, de Atributos, de Texto, etc.
- Puede ser Static o Live depende como se obtenga
- Será Static, si la Node List se obtuvo con el método `document.querySelectorAll(selector)``
- Sera Live, si la Node List se obtuvo de la propiedad `elemento.ChildNodes`
- Es un array-like, no tiene los métodos de los arreglos, pero tiene propiedad length y accesamos a los elementos con corchetes [n]
- Contiene los siguientes métodos:
  - `forEach(()=>{})` Tiene el metodo foreach como los arreglod que ejecuta la función especificada en cada nodo
  - `item(n)` Regresa el elemento en el índice n que se le especifique
  - `keys()` Regresa las llaves
  - `values()` Regresa los valores
  - `entries()` Regresa los pares llave-valor

DOM API (Iteración en una HTML Collection)

```
let collection = document.getElementsByClassName('clase');  // Obtenemos una Html Collection de Elementos del DOM

for (let i=0; i<collection.length; i++) {                   // Recorremos con un Ciclo for clasico
        console.log(collection[i]);                         // También pueden usarse ciclos while y dowhile
}

for (let i of collection) {                                 // Recorremos con un Ciclo for of
        console.log(collection[i]);
}

Array.from(collection).forEach(function (elemento) {        // Convertir la Html Collecion en array y usar el método
forEach()
        console.log(elemento)
});
```

> **El ciclo for-of** nos permite recorrer objetos iterables (como las cadenas, arreglos, Html Collections y Node Lists)
> **El método Array.from(array-like)** de los arreglos nos permite crear un arreglo de un objeto array-like o de un objeto iterable

DOM API (Iteración en una Node List)

```
let nodelist = document.querySelectorAll('CssSelector');  // Obtenemos una Node List de Elementos del DOM

for (let i=0; i<NodeList.length; i++) {                   // Recorremos con un ciclo for clasico
        console.log(nodelist[i]);                         // También pueden usarse ciclos while y dowhile
}

nodelist.forEach(function(elemento) {                     // con un ciclo forEach
      console.log(elemento);
});

for (let elemento of nodelist) {                          // Ciclo for of
    console.log(elemento);
}

Array.from(nodelist).forEach(function (elemento) {        // Convirtiendo la Node List en un array y usando el método
forEach()
        console.log(elemento)
});
```

DOM API

Obtener Padre e Hijos

Método DOM API	**Regresa...**
`document.getElementById('id').parentNode`	El Padre
`document.getElementById('id').childNodes`	Los hijos

Contenido del elemento

Método DOM API	**Obtiene....**
`document.getElementById('h1').innerHTML`	Contenido incluyendo etiquetas html si tiene
`document.getElementById('h1').innerText`	Contenido sin espacios (si tiene)
`document.getElementById('h1').textContent`	Contenido como este (aún con espacios)

Búsqueda

Método DOM API	**Que hace?**	**Regresa**
`elemento.matches('Selector')`	Verifica si elemento cumple el selector dado	true/false
`elemento.closest('selector')`	Navega hacia arriba en el DOM buscando el elemento que cumpla con el selector dato	elemento/null

DOM API

Trabajar con Clases de Css (agregar, eliminar, etc)

Método DOM API	**Que hace?**
`document.getElementById('id').classList.remove('clase')`	Elimina la clase `clase` del elemento con el id `id`
`document.getElementById('id').classList.add('clase')`	Agrega la clase `clase` del elemento con el id `id`
`document.getElementById('id').classList.toggle('clase')`	Si la `clase` existe la agrega, si no existe la elimina
`document.getElementById('id').classList.contains('clase')`	Regresa Verdadera/Falso si el elemento contiene la `clase`

DOM API

Ejemplo de Creación y Agregación de Elementos

```
var encabezado = document.createElement('h1');                    // Crear el Elemento
var contenido  = document.createTextNode('Esto es un encabezado'); // Crear el Contenido
encabezado.appendChild(encabezadoCont);                           // Agregar el contenido al elemento

var contenedor = document.getElementById('#contenedor');          // Obtener el elemento al que le vamos a agregar algo
contenedor.appendChild(encabezado);                               // Agregar el nuevo elemento como hijo
contenedor.insertBefore(encabezado);                              // Agregar nuevo elmento "antes de"
```

Ejemplo de Inserción de Html

```
var contenedor = document.getElementById('#contenedor');
contenedor.insertAdjacentHTML('beforebegin', '<div><p>Esto es un div con un parrafo</p></div>');
// beforebegin -> Antes del elemento (como hermano)
// afterbegin  -> Adentro del elemento (como primer hijo)
// beforeend   -> Adentro del elemento (despues del ultimo hijo)
// afterend    -> Despues del elemento (como hermano)
```