cedia | TMO Training Management Office

info@cedia.org.ec
cedia.edu.ec

CURSO DE JAVASCRIPT

Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia

Ing. María Fernanda Granda J., PhD

2-12-2025

"Conectando ideas, transformando sociedades"

Módulo 7: Trabajar con el Modelo Objeto Documento (DOM)

- Acceso al DOM-HTML.
- Modificación del DOM-HTML.
- Incompatibilidades de los navegadores y código cross-browser.

Ejemplo 5

- Lista del supermercado con prompt()
- **Objetivo**: Usar el DOM + prompt() para agregar ítems a una lista evitando duplicados.
- Pista usar: includes

Ejemplo 5

- Una posible solución

```
function agregarArticulo() {
    let nuevo = prompt("Ingresa un artículo para comprar:");
    if (!nuevo) return;

    // Obtener todos los ítems actuales
    let items = Array.from(document.querySelectorAll("#superLista li")).map(li => li.textContent.toLowerCase());

    if (items.includes(nuevo.toLowerCase())) {
      alert("⚠️ Ese artículo ya está en la lista.");
    } else {
      let li = document.createElement("li");
      li.textContent = nuevo; // el nuevo artículo
      document.getElementById("superLista").appendChild(li);
      alert("✅ Artículo agregado. Lista actualizada.");
    }
  }
```

Ejemplo 5

- Una posible solución

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo Supermercado</title>
    <link rel="stylesheet" href="estilo.css">
</head>
<body>
    <script src="super.js"></script>
    <br>
    <p>Lista de Items</p>
    <ul id="superLista"></ul>
    <li>Computadora</li>
    <li>Impresora</li>
    <li>Monitor</li>
    </ul>
    <button onclick="agregarArticulo()">Agregar Artículo</button>
</body>
</html>
```

Usando los métodos getElementsBy

- La API DOM tiene métodos que utilizan selectores similares a CSS para buscar y seleccionar elementos HTML.
  - **querySelector** devuelve el primer elemento que coincide con el selector.
  - **querySelectorAll** devuelve una colección de todos los elementos que coinciden con el selector.
  - No compatible con navegadores anteriores (anteriores a IE 8).
- Ambos métodos toman como parámetro de cadena el selector CSS del elemento.

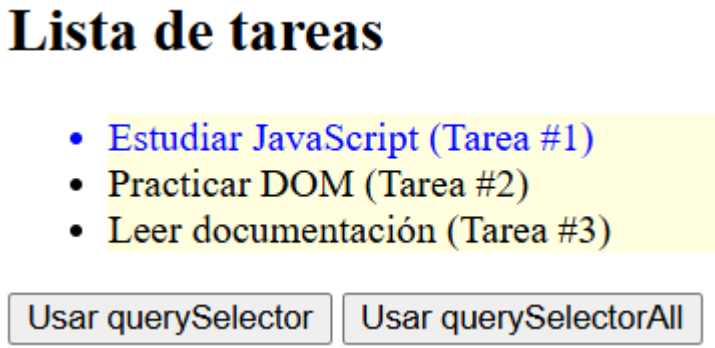
```
var header = document.querySelector('#header');
//the element with id="header"
var navItems = document.querySelectorAll('#main-nav li');
//li elements contained in element with id=main-nav
```

Ejemplo

Lista de tareas

- Estudiar JavaScript
- Practicar DOM
- Leer documentación

Usar querySelector | Usar querySelectorAll



Ejemplo

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>querySelector vs querySelectorAll</title>
</head>
<body>
  <h2>Lista de tareas</h2>
  <ul>
    <li class="tarea">Estudiar JavaScript</li>
    <li class="tarea">Practicar DOM</li>
    <li class="tarea">Leer documentación</li>
  </ul>

  <button onclick="usarQuerySelector()">Usar querySelector</button>
  <button onclick="usarQuerySelectorAll()">Usar querySelectorAll</button>
  <script src="otro-codigo.js">  </script>
</body>
</html>
```

```js
function usarQuerySelector() {
  // Selecciona SOLO el primer elemento con la clase "tarea"
  let primeraTarea = document.querySelector(".tarea");
  primeraTarea.style.color = "blue";
  alert("querySelector seleccionó: " + primeraTarea.textContent);
}

function usarQuerySelectorAll() {
  // Selecciona TODOS los elementos con la clase "tarea"
  let todasLasTareas = document.querySelectorAll(".tarea");
  todasLasTareas.forEach((item, index) => {
    item.style.backgroundColor = "lightyellow";
    item.textContent += ` (Tarea #${index + 1})`;
  });
  alert(`querySelectorAll modificó ${todasLasTareas.length} tareas`);
}
```

Seleccionando elementos dentro de otros elementos

- La API DOM permite seleccionar elementos dentro de otros elementos.
  - Selecciona todos los divs dentro de un elemento con el id "wrapper".

```
var wrapper = document.getElementById('wrapper');
var divsInWrapper = wrapper.getElementsByTagName('div');
```

  - Todos los métodos se pueden utilizar en elementos HTML, excepto getElementById().

NodeLists

- Una lista de nodos es una colección devuelta por los métodos del selector de DOM:
  - getElementsByTagName(tagName)
  - getElementsByName(name)
  - getElementsByClassName(className)
  - querySelectorAll(selector)

```
var divs = document.getElementsByTagName('div');
var queryDivs = document.querySelectorAll('div');
for(var i=0; i< divs.length; i++){
   //do stuff with divs[i]…
}
```

NodeLists

- NodeList parece un array, pero no lo es.
  - Es un objeto con propiedades similares a un array.
    - Tiene longitud e indexador.
  - Recorrer un array con un bucle for-in funciona de forma inesperada.

```
for (var i in divs) {
   console.log('divs[' + i + '] = ' + divs[i]);
}
//divs[0] = ...
//divs[1] = ...
//divs[length] = ...
//divs[item] = ...
```

Ejemplo de uso getElementsByTagName

- Cambiar el contenido de todos los párrafos (<p>)

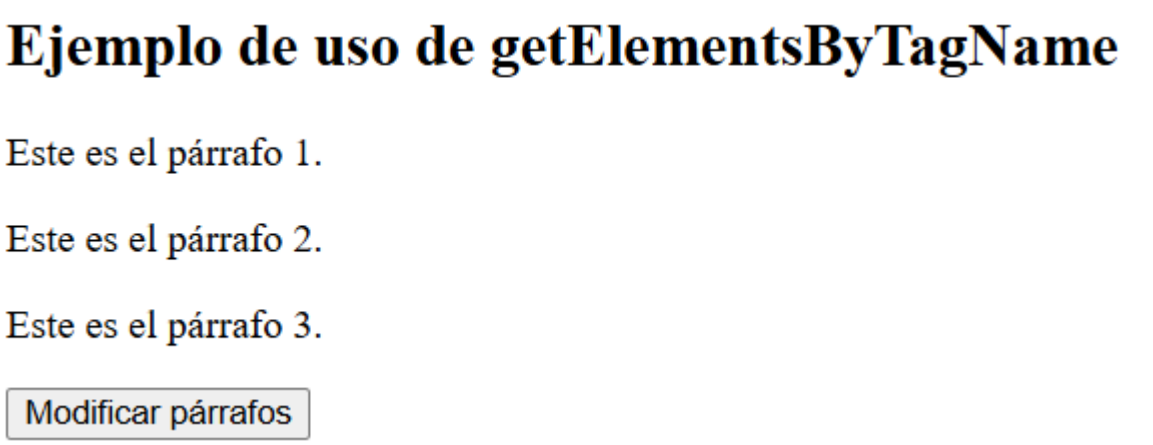
```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>getElementsByTagName</title>
</head>
<body>
  <h2>Ejemplo de uso de getElementsByTagName</h2>
  <p>Este es el párrafo 1.</p>
  <p>Este es el párrafo 2.</p>
  <p>Este es el párrafo 3.</p>
  <button onclick="modificarParrafos()">Modificar párrafos</button>
  <script src="codigo.js"></script>
</body>
</html>
```

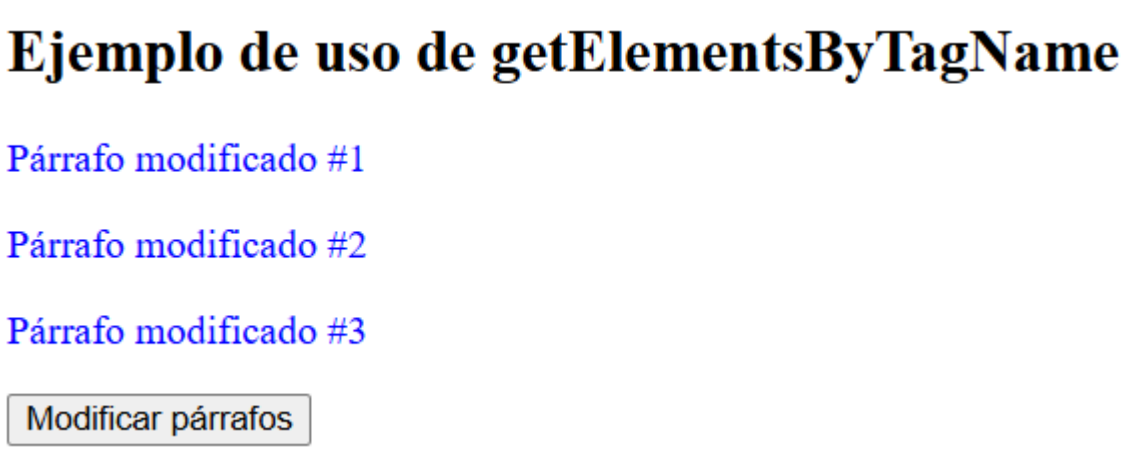
```javascript
function modificarParrafos() {
  // Obtener todos los elementos <p>
  let parrafos = document.getElementsByTagName("p");

  // Recorrerlos y modificar su contenido
  for (let i = 0; i < parrafos.length; i++) {
    parrafos[i].textContent = `Párrafo modificado #${i + 1}`;
    parrafos[i].style.color = "blue";
  }
}
```

Ejemplo de uso getElementsByTagName

- Cambiar el contenido de todos los párrafos (<p>)





Listas de nodos estáticas y activas

- Hay dos tipos de NodeLists:
  - getElementsBy...() devuelve una LiveNodeList
  - querySelectorAll() devuelve una StaticNodeList
- Las LiveLists registran los elementos seleccionados,
  - incluso cuando se realizan cambios en el DOM.
  - Mientras que las listas estáticas contienen los elementos tal como estaban al ejecutar el método.
- Ten en cuenta que LiveNodeList es más lento que un array normal.
  - Es necesario almacenar en caché su longitud para un mejor rendimiento

¿Cómo funciona el DOM en JavaScript?

- **1. El Objeto document:** El DOM se accede a través del objeto **document**, que es el objeto principal que representa la página web.
- **2. Localización de Elementos:** Los scripts pueden encontrar elementos específicos en la página usando selectores CSS o métodos del DOM, como:
  - getElementById("id")
  - getElementsByClassName("clase")
  - getElementsByTagName("etiqueta")
  - querySelector("selector") / querySelectorAll("selector")

¿Cómo funciona el DOM en JavaScript?

- **3. Modificación de Elementos:** Una vez que un script ha localizado un elemento, puede modificar sus propiedades, estilos, contenido, atributos, etc.
- **4. Manipulación del Árbol:** Los scripts pueden crear, insertar, eliminar o reemplazar nodos en el árbol DOM, cambiando así la estructura de la página.
- **5. Eventos:** El DOM permite a los scripts responder a eventos (clics, cambios de entrada, etc.) que ocurren en la página, ejecutando código en respuesta a estos eventos.

Manejo de Eventos de HTML desde JS

- Se realiza principalmente:
  - usando el método addEventListener() para asociar una función de callback a un evento específico en un elemento del DOM,
  - asignando directamente la función a propiedades como onclick.
- Se recomienda addEventListener() porque permite agregar múltiples escuchas de eventos, a diferencia de la asignación directa de propiedades que solo admite una.

Ejemplo usando el método addEventListener()

```
const btnAgregar = document.getEementById("btnAgregar");
```

```
// Evento: Agregar tarea
btnAgregar.addEventListener("click", () => {
    const texto = entrada.value.trim();

}
```

Ejemplo usando el método addEventListener()

- Se puede pasar también un objeto.

```
let button1 = document.getElementById("button1");


let obj = {
    handleEvent(event){
        alert(event.type);
    }
}


button1.addEventListener("click", obj);
```

Ejemplo usando el método addEventListener()

- Se puede manejar varios eventos

```
let button1 = document.getElementById("button1");


let obj = {
    handleEvent(event) {
        switch (event.type) {
            case "click":
                alert("Hola");
                break;
            case "mouseup":
                alert("Adios");
                break;
        }
    }
}


button1.addEventListener("click", obj);
button1.addEventListener("mouseup", obj);
```

Asignación directa de propiedades

- Consiste en asignar una función directamente a una propiedad del elemento que corresponda al evento.
- **Limitación:** Un elemento solo puede tener una función asignada a una propiedad específica, lo que impide agregar múltiples manejadores de eventos para el mismo evento de esta manera.

```
<input type="text" id="tarea" placeholder="Escribe una tarea...">
<button id="btnAgregar" onclick="Agregar()">Agregar Tarea</button>
```

```
const btnAgregar = document.querySelector('#btnAgregar');

btnAgregar.onclick = function() {
    alert('¡Hola desde JavaScript!');
};
```

```
function hello(){
    return "Hola";
}


button1.onclick = hello;
```

cedia | TMO Training Management Office

GRACIAS

cedia.edu.ec
info@cedia.org.ec
(+593) 7-407-9300