



HTML5 - CSS3 - JS6 – DOM



HTML



CSS



HTML5, CSS3 y ES6 DOM

Centro público integrado de
formación profesional
Alan Turing

JS





Índice

¿Qué veremos en esta sección?

Bloque 1 – Primer ejercicio: getElementById()

Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

Bloque 3 – querySelector() y querySelectorAll()

Bloque 4 – textContent, innerText e innerHTML

Bloque 5 – createElement(), appendChild() y removeChild()

Bloque 6 – addEventListener() y removeEventListener()

Bloque 7 – Navegación por el árbol del DOM



HTML5 - CSS3 - JS6 – DOM



Bloque 1 – Primer ejercicio: `getElementById()`

Bloque 1 – Primer ejercicio: `getElementById()`



Bloque 1 – Primer ejercicio: getElementById()

Objetivo: Creamos nuestro primer ejercicio de JavaScript. Aprenderemos a conectar HTML y JS correctamente, a detectar elementos de la página con getElementById() y a responder a eventos como un clic.

1) Paso 1: Crea la estructura de carpetas

Dentro del proyecto principal, sitúate en la carpeta /ejercicios y crea esta estructura en VS Code como las veces anteriores.

Estructura final:

ejercicios

```
  └── css
      └── styles.css      ← Estilos comunes para las tarjetas
    └── ejer01
        ├── index.html    ← HTML del ejercicio
        └── js
            └── script.js   ← Lógica del ejercicio
        └── css
            └── (vacío, reservado para estilos propios futuros)
```

12
34

HTML5 - CSS3 - JS6 – DOM



BLOQUE 1 – getElementById()

2) Paso 2: Código del index.html del ejercicio. Copia este contenido en ejercicios/ejer01/index.html: Este archivo es la estructura visual del ejercicio. Incluye un botón y un párrafo donde se mostrará el resultado:

index.html (ejer01)

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Ejercicio 01 - getElementById()</title>
    <link rel="icon" href="../../img/favicon.ico" type="image/x-icon" />
    <!-- Estilos generales del sitio -->
    <link rel="stylesheet" href="../../css/styles.css" />
    <!-- Estilos comunes de ejercicios (tarjetas) -->
    <link rel="stylesheet" href="../css/styles.css" />
    <!-- Lógica JS del ejercicio -->
    <script defer src=".js/script.js"></script>
</head>

<body>
    <header class="header">
        <div class="header-content">
            <h3>Ejemplos - FUNDAMENTOS DE JS (ES6)</h3>
            <h4>Desarrollo WEB en entorno cliente</h4>
            <h2>Ejercicio 01 - getElementById()</h2>
        </div>
        <div class="header-profile">
            
        </div>
    </header>
```

```
<nav>
    <ul class="navbar">
        <li><a href="https://www.w3schools.com/js/default.asp">w3schools</a></li>
        <li><a href="https://es.javascript.info/js">javascript.info</a></li>
        <li><a href="https://developer.mozilla.org/es/docs/Web/JavaScript">MDN</a></li>
    </ul>
</nav>

<main>
    <div class="container">
        <div class="card">
            <div class="cardTitle">Uso básico de getElementById()</div>
            <div class="cardBody">
                <input id="boton" type="button" value="Púlsame">
                <p id="resultado">(Aqui se mostrará el resultado del ejercicio)</p>
            </div>
            <div class="cardFooter">
                <a href="../../index.html">&gt;&gt; Página principal</a>
            </div>
        </div>
    </main>

    <footer class="footer">
        © José García ©
    </footer>
</body>
</html>
```

12
34

HTML5 - CSS3 - JS6 – DOM



BLOQUE 1 – getElementById()

Paso 3: Código JavaScript – script.js. Abre ejercicios/ejer01/js/script.js y copia el siguiente código:

```
"use strict";

// 1. Captura los elementos del DOM usando getElementById
const boton = document.getElementById("boton");
const resultado = document.getElementById("resultado");

// 2. Asigna una función al evento "click"
boton.onclick = mostrar;

// 3. Define la función que se ejecutará cuando se pulse el botón
function mostrar() {
    resultado.innerHTML = "¡Has pulsado el botón correctamente!";
}
```

12
34

BLOQUE 1 – getElementById()

HTML5 - CSS3 - JS6 – DOM



Paso 4: Entender la vinculación HTML ↔ JS

Elemento

```
<script defer src=".js/script.js">
```

id="boton" y id="resultado"

```
getElementById("boton")
```

Explicación

Carga el JS solo después de que todo el HTML esté cargado. Impide errores por elementos que aún no existen.

Son identificadores únicos del DOM para que JS pueda encontrarlos fácilmente.

Busca el botón y lo guarda en una variable para poder asignarle eventos.

¿Qué es el DOM?

El DOM (Document Object Model) es una representación de la página web como un árbol de nodos. Cada etiqueta HTML se convierte en un nodo que JavaScript puede leer, modificar, eliminar o crear.

12
34

HTML5 - CSS3 - JS6 – DOM



BLOQUE 1 – getElementById()

¿Qué hace **getElementById()**?

- **Busca** un elemento concreto del HTML que tenga un **id** específico.
- Devuelve una **referencia** al objeto HTML para poder manipularlo desde JS.

Ejemplo:

```
const resultado = document.getElementById("resultado");
resultado.innerHTML = "Texto nuevo";
```

🔍 Este código:

- Encuentra el elemento `<p id="resultado">...</p>`
- **Cambia** su **contenido** interno (`innerHTML`) por el texto "Texto nuevo"

12
34

HTML5 - CSS3 - JS6 – DOM

BLOQUE 1 – getElementById()



Resultado esperado

Ejemplos – FUNDAMENTOS DE JS (ES6)
Desarrollo WEB en entorno cliente

Ejercicio 01 – getElementById()

w3schools javascript.info MDN



Uso básico de getElementById()

(Aquí se mostrará el resultado del ejercicio)

[>> Página principal](#)

Ejemplos – FUNDAMENTOS DE JS (ES6)

Desarrollo WEB en entorno cliente

Ejercicio 01 – getElementById()



w3schools javascript.info MDN

Uso básico de getElementById()

¡Has pulsado el botón correctamente!

[>> Página principal](#)

© José García ©



HTML5 - CSS3 - JS6 – DOM

A

**Bloque 2 – Ejercicio 02: Selección con
getElementsByClassName() y getElementsByTagName()**

Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()



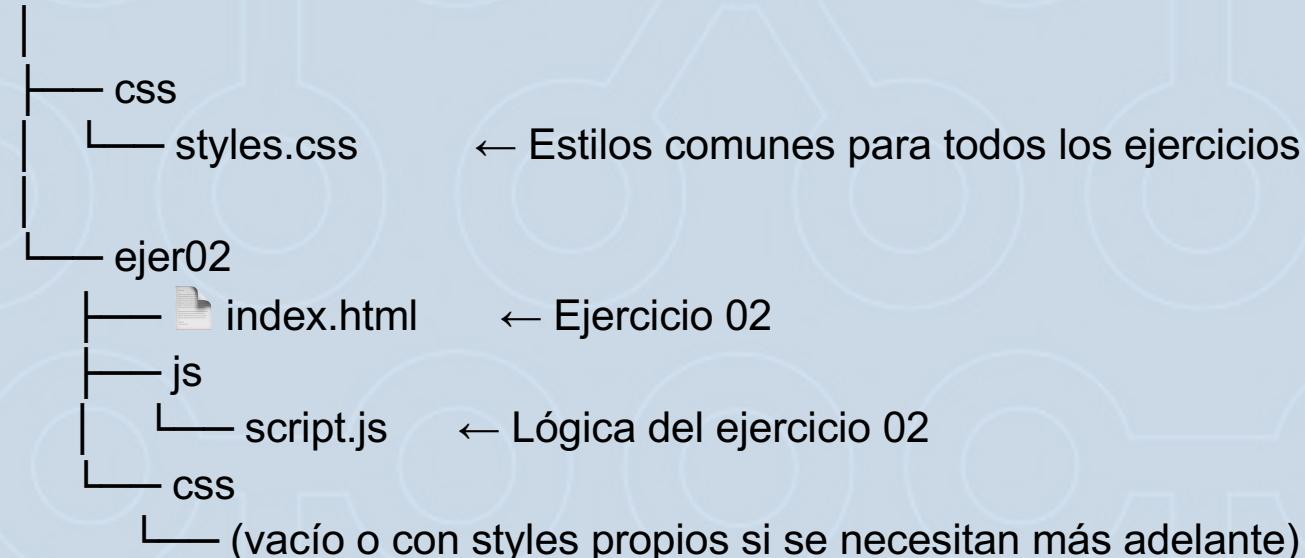
Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

Objetivo: Aprender a seleccionar varios elementos del DOM usando `getElementsByClassName()` y `getElementsByTagName()`. Comparar ambos métodos y entender cuándo conviene usar cada uno.

1) Paso 1: Creación de archivos y carpetas

Dentro de la carpeta ejercicios, crea la siguiente estructura:

ejercicios





Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

2) Paso 2: Código HTML (ejer02/index.html)

Solo cambia el contenido del <main>. El resto del HTML es exactamente igual al del ejer01.

index.html (ejer02, solo modificamos el main)	
<pre><main> <div class="container"> <div class="card"> <div class="cardTitle">Selección con getElementByClassName() y getElementsByClassName()</div> <div class="CardBody"> <button id="btnClase">Resaltar párrafos (por clase)</button> <button id="btnEtiqueta">Resaltar títulos (por etiqueta)</button> <h3>Sección A</h3> <p class="texto">Primer párrafo</p> <p class="texto">Segundo párrafo</p></pre>	<pre><h3>Sección B</h3> <p class="texto">Tercer párrafo</p> <p class="texto">Cuarto párrafo</p> </div> <div class="cardFooter"> &gt;&gt; Página principal </div> </div> </main></pre>



Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

3) Paso 3: Código JavaScript (ejer02/js/script.js)

script.js	
<pre>"use strict"; // 1. Capturamos los elementos del DOM const botonClase = document.getElementById("btnClase"); const botonEtiqueta = document.getElementById("btnEtiqueta"); // 2. Asignamos funciones a los botones botonClase.onclick = resaltarParrafos; botonEtiqueta.onclick = resaltarTitulos; // 3. Función para resaltar los párrafos usando getElementsByTagName() function resaltarParrafos() { const parrafos = document.getElementsByClassName("texto"); for (let i = 0; i < parrafos.length; i++) { parrafos[i].style.backgroundColor = "#ffffcc"; parrafos[i].style.border = "1px solid #ccc"; parrafos[i].style.padding = "5px"; } }</pre>	<pre>// 4. Función para resaltar los títulos usando getElementsByTagName() function resaltarTitulos() { const titulos = document.getElementsByTagName("h3"); for (let i = 0; i < titulos.length; i++) { titulos[i].style.color = "darkred"; titulos[i].style.textDecoration = "underline"; } }</pre>



Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

4) Explicación didáctica (para que les hables tú directamente a tus alumnos)

Chicos, fijaos en lo siguiente:

En el ejer01 usábamos getElementById("...") para seleccionar un único elemento concreto.

Eso es muy útil cuando solo hay un botón, una caja, un párrafo concreto...

Pero ahora tenemos varios párrafos y varios títulos.

Aquí entran en juego otros métodos del DOM:

getElementsBy**Class**Name("...")

- Devuelve todos los elementos que tengan esa clase.
- El resultado es una colección HTML (**HTMLCollection**), parecida a un **array**.
- En nuestro ejemplo:

`document.getElementsByClassName("texto")` selecciona los 4 párrafos que comparten esa clase.

Es útil cuando queremos aplicar un cambio a varios elementos que comparten un mismo estilo o propósito.

getElementsBy**Tag**Name("...")

- Devuelve todos los elementos de una etiqueta concreta (como p, h3, img...).
- En nuestro ejemplo:
- `document.getElementsByTagName("h3")` selecciona los dos títulos de sección.

Es útil cuando queremos afectar a todos los elementos de un tipo específico, sin importar su clase.



Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

Diferencia clave:

Método	¿Qué selecciona?	¿Cuándo usarlo?
getElementById("...")	Un único elemento con un ID	Cuando solo hay uno, y es único
getElementsByClassName("...")	Todos los elementos con esa clase	Cuando varios elementos comparten el mismo estilo/clase
getElementsByTagName("...")	Todos los elementos de una etiqueta (p, h3, etc)	Cuando queremos afectar a todos los elementos del mismo tipo



Bloque 2 – Ejercicio 02: Selección con getElementsByClassName() y getElementsByTagName()

Resultado esperado

Al pulsar los botones del ejercicio:

Se resaltan todos los párrafos con clase texto con fondo amarillo claro y borde gris.

Se resaltan todos los títulos <h3> con color rojo oscuro y subrayado.

**Selección con getElementsByClassName() y
getElementsByTagName()**

Sección A

Primer párrafo

Segundo párrafo

Sección B

Tercer párrafo

Cuarto párrafo

[>> Página principal](#)

**Selección con getElementsByClassName() y
getElementsByTagName()**

Sección A

Primer párrafo

Segundo párrafo

Sección B

Tercer párrafo

Cuarto párrafo

[>> Página principal](#)

**Selección con getElementsByClassName() y
getElementsByTagName()**

Sección A

Primer párrafo

Segundo párrafo

Sección B

Tercer párrafo

Cuarto párrafo

[>> Página principal](#)

José García

DOM



HTML5 - CSS3 - JS6 – DOM



Bloque 3 – `querySelector()` y `querySelectorAll()`

Bloque 3 – `querySelector()` y `querySelectorAll()`



Bloque 3 – querySelector() y querySelectorAll()

Objetivo. Aprender a seleccionar elementos en el DOM usando querySelector() y querySelectorAll(), dos métodos más flexibles que permiten usar selectores de CSS. 1) Paso 1: Creación de archivos y carpetas

Paso 1 – Preparar el HTML

El archivo base es el mismo que en los ejercicios anteriores. Solo modificamos el contenido del <main> en index.html:

ejer03/index.html	
<pre><main> <div class="container"> <div class="card"> <div class="cardTitle">Selección con querySelector() y querySelectorAll()</div> <div class="CardBody"> <button id="btnSelector">Usar querySelector()</button> <button id="btnSelectorAll">Usar querySelectorAll()</button> <h3>Ejemplo A</h3> <p class="parrafo">Soy el primer párrafo.</p> <p class="parrafo">Soy el segundo párrafo.</p></pre>	<pre><h3>Ejemplo B</h3> <p id="especial">Soy un párrafo especial con ID.</p> <div id="resultado"></div> <div class="cardFooter"> &gt; &gt; Página principal </div> </div> </main></pre>



Bloque 3 – querySelector() y querySelectorAll()

Paso 2 – Preparar el JS

ejer03/js/script.js

```
"use strict"; // Modo estricto recomendado

// Capturamos los botones y el div de resultado
const btnSelector = document.getElementById("btnSelector");
const btnSelectorAll =
document.getElementById("btnSelectorAll");
const resultado = document.getElementById("resultado");

// Ejemplo con querySelector()
btnSelector.addEventListener("click", () => {
    // Selecciona el PRIMER elemento que cumpla con el selector
    // CSS
    const primerParrafo = document.querySelector(".parrafo");

    // Mostramos el texto en el resultado
    resultado.innerHTML = `<p>querySelector() encontró:
<b>${primerParrafo.textContent}</b></p>`;
});
```

```
// Ejemplo con querySelectorAll()
btnSelectorAll.addEventListener("click", () => {
    // Selecciona TODOS los elementos que cumplan el selector
    // CSS
    const todosParrafos =
document.querySelectorAll(".parrafo");

    // Recorremos la NodeList con forEach
    let salida = "<p>querySelectorAll() encontró:</p><ul>";
    todosParrafos.forEach((p) => {
        salida += `<li>${p.textContent}</li>`;
    });
    salida += "</ul>";

    resultado.innerHTML = salida;
});
```



Bloque 3 – querySelector() y querySelectorAll()

- Paso 3 – Diferencia entre querySelector y querySelectorAll

Método	Qué devuelve	Ejemplo
querySelector()	Solo el primer elemento que cumple el selector CSS.	document.querySelector(".parrafo") → selecciona el primer párrafo con clase .parrafo.
querySelectorAll()	Todos los elementos que cumplen el selector CSS, en una NodeList .	document.querySelectorAll(".parrafo") → selecciona todos los párrafos con clase .parrafo.

👉 Importante: NodeList ≠ Array, aunque se parecen, pero sí podemos recorrerlo con forEach.



Bloque 3 – querySelector() y querySelectorAll()

Paso 4 – Resultado esperado

Al pulsar “Usar querySelector()”, en la caja de resultado aparece el texto del primer párrafo con clase .parrafo.

Al pulsar “Usar querySelectorAll()”, en la caja de resultado aparece una lista con el contenido de todos los párrafos con clase .parrafo.

Selección con querySelector() y querySelectorAll()

Ejemplo A

Soy el primer párrafo.
Soy el segundo párrafo.

Ejemplo B

Soy un párrafo especial con ID.

PÁRRAFOS ENCONTRADOS:

[>> Página principal](#)

Selección con querySelector() y querySelectorAll()

Ejemplo A

Soy el primer párrafo.
Soy el segundo párrafo.

Ejemplo B

Soy un párrafo especial con ID.

PÁRRAFOS ENCONTRADOS:

querySelector() encontró: **Soy el primer párrafo.**

[>> Página principal](#)

Selección con querySelector() y querySelectorAll()

Ejemplo A

Soy el primer párrafo.
Soy el segundo párrafo.

Ejemplo B

Soy un párrafo especial con ID.

PÁRRAFOS ENCONTRADOS:

- querySelectorAll() encontró:
- Soy el primer párrafo.
 - Soy el segundo párrafo.

[>> Página principal](#)



HTML5 - CSS3 - JS6 – DOM



Bloque 4 – textContent, innerText e innerHTML

Bloque 4 – textContent, innerText e innerHTML



Bloque 4 – textContent, innerText e innerHTML

Objetivo. Aprender a leer y modificar el contenido de los elementos HTML mediante tres propiedades:

- `textContent`: devuelve todo el texto, incluso el oculto.
- `innerText`: devuelve el texto visible en pantalla.
- `innerHTML`: devuelve todo el HTML interno, incluyendo etiquetas.

Paso 1 – Preparar el HTML

ejer04/index.html

```
<main>
  <div class="container">
    <div class="card">
      <div class="cardTitle">Leer y modificar contenido con textContent,
      innerText e innerHTML</div>
      <div class="CardBody">
        <button id="btnLeer">Leer contenido</button>
        <button id="btnModificar">Modificar contenido</button>

        <div id="caja">
          <h3>Mensaje original</h3>
          <p class="parrafo">Este <b>texto</b> tiene <span
          style="display:none">contenido oculto</span> y <i>HTML</i>.</p>
        </div>
```

```
<div id="resultado"></div>
  </div>
  <div class="cardFooter">
    <a href="../../index.html">&gt;&gt; Página principal</a>
  </div>
</div>
</main>
```



Bloque 4 – textContent, innerText e innerHTML

Paso 2 – Preparar el JS

ejer04/js/script.js	
<pre>"use strict"; // Activamos modo estricto // Capturas iniciales const btnLeer = document.getElementById("btnLeer"); const btnModificar = document.getElementById("btnModificar"); const caja = document.getElementById("caja"); const parrafo = caja.querySelector(".parrafo"); const resultado = document.getElementById("resultado"); // 🔍 Leer contenido usando las tres propiedades btnLeer.addEventListener("click", () => { const texto1 = parrafo.textContent; const texto2 = parrafo.innerText; const texto3 = parrafo.innerHTML;</pre>	<pre>resultado.innerHTML = ` <p>textContent: \${texto1}</p> <p>innerText: \${texto2}</p> <p>innerHTML: \${texto3}</p> `; }); // 🚀 Modificar el contenido usando innerHTML btnModificar.addEventListener("click", () => { parrafo.innerHTML = `Texto <mark>modificado</mark> usando <code>innerHTML</code>.`; resultado.innerHTML = `<p>✅ Contenido modificado correctamente.</p>`; });</pre>



Bloque 4 – textContent, innerText e innerHTML

Paso 3 – Diferencias entre las tres propiedades

Propiedad	¿Qué devuelve?	¿Incluye HTML?	¿Muestra contenido oculto?
textContent	Todo el texto	No	Sí
innerText	Solo lo visible	No	No
innerHTML	Código HTML	Sí	Sí

Usa textContent si solo necesitas texto plano.

Usa innerHTML si quieres modificar etiquetas.



Bloque 4 – textContent, innerText e innerHTML

Paso 4 – Resultado esperado

1. Al pulsar “Leer contenido”, se muestra el contenido del párrafo usando las tres propiedades.
2. Al pulsar “Modificar contenido”, se sobrescribe el contenido del párrafo con un nuevo HTML.

Leer y modificar contenido con textContent, innerText e innerHTML

[Leer contenido](#) [Modificar contenido](#)

Mensaje original

Este **texto** tiene y *HTML*.

Leer y modificar contenido con textContent, innerText e innerHTML

[Leer contenido](#) [Modificar contenido](#)

Mensaje original

Este **texto** tiene y *HTML*.

textContent: Este texto tiene contenido oculto y HTML.

innerText: Este texto tiene y HTML.

innerHTML: Este **texto** tiene y *HTML*.

[>> Página principal](#)

Leer y modificar contenido con textContent, innerText e innerHTML

[Leer contenido](#) [Modificar contenido](#)

Mensaje original

Texto **modificado** usando innerHTML.

Contenido modificado correctamente.

[>> Página principal](#)



Bloque 5 – createElement(), appendChild() y removeChild()

Bloque 5 – createElement(), appendChild() y removeChild()



Bloque 5 – createElement(), appendChild() y removeChild()

Objetivo. Aprender a crear, insertar y eliminar elementos del DOM de forma dinámica.

Estos métodos permiten construir interfaces de forma programada sin modificar directamente el HTML.

Paso 1 – Preparar el HTML

ejer05/index.html

```
<main>
  <div class="container">
    <div class="card">
      <div class="cardTitle">Crear y eliminar elementos
dinámicamente</div>
      <div class="CardBody">
        <button id="btnCrear">Crear nuevo elemento</button>
        <button id="btnEliminar">Eliminar último
elemento</button>

        <ul id="lista">
          <li>Elemento inicial 1</li>
          <li>Elemento inicial 2</li>
        </ul>
      </div>
    </div>
  </div>
</main>
```

```
<div id="resultado"></div>
  </div>
  <div class="cardFooter">
    <a href="../../index.html">&gt;&gt; Página principal</a>
  </div>
</div>
</main>
```



Bloque 5 – createElement(), appendChild() y removeChild()

Paso 2 – Preparar el JS

ejer05/js/script.js

```
"use strict"; // Modo estricto activado

// Captura de elementos
const btnCrear = document.getElementById("btnCrear");
const btnEliminar = document.getElementById("btnEliminar");
const lista = document.getElementById("lista");
const resultado = document.getElementById("resultado");

// Crear un nuevo <li> y añadirlo al final de la lista
btnCrear.addEventListener("click", () => {
  const nuevoElemento = document.createElement("li"); // Creamos un nuevo <li>
  nuevoElemento.textContent = "Nuevo elemento creado"; // Le damos contenido
  lista.appendChild(nuevoElemento); // Lo añadimos al final de la lista

  resultado.innerHTML = "<p>✓ Elemento creado y añadido a la lista.</p>";
});
```

```
// Eliminar el último <li> si hay más de 0
btnEliminar.addEventListener("click", () => {
  if (lista.children.length > 0) {
    const ultimo = lista.lastElementChild; // Seleccionamos el último <li>
    lista.removeChild(ultimo); // Lo eliminamos
    resultado.innerHTML = "<p>☒ Último elemento eliminado.</p>";
  } else {
    resultado.innerHTML = "<p>! No hay elementos que eliminar.</p>";
  }
});
```



Bloque 5 – createElement(), appendChild() y removeChild()

Paso 3 – Explicación didáctica de los métodos del DOM

Método	¿Para qué sirve?
createElement()	Crea un nuevo nodo HTML (aún no está en pantalla). Ej: document.createElement("li")
appendChild()	Añade un nodo hijo al final de otro. Ej: lista.appendChild(nodo)
removeChild()	Elimina un nodo hijo de un nodo padre. Ej: lista.removeChild(nodo)

Importante:

- Siempre que uses removeChild() necesitas especificar qué hijo eliminar.
- Puedes usar lastElementChild para obtener el último hijo de un elemento.
- Estos métodos funcionan con nodos reales del DOM (no con texto o strings HTML).



Bloque 5 – createElement(), appendChild() y removeChild()

Paso 4 – Resultado esperado

1. Al pulsar “Crear nuevo elemento”, se añade un con el texto “Nuevo elemento creado” al final de la lista.
2. Al pulsar “Eliminar último elemento”, se elimina el último de la lista, si existe.
3. Si la lista queda vacía, aparece un mensaje de advertencia.

Crear y eliminar elementos dinámicamente

[Crear nuevo elemento](#) [Eliminar último elemento](#)

- Elemento inicial 1
- Elemento inicial 2

Crear y eliminar elementos dinámicos

[Crear nuevo elemento](#) [Eliminar último elemento](#)

- Elemento inicial 1
- Elemento inicial 2
- Nuevo elemento creado
- Nuevo elemento creado
- Nuevo elemento creado

Elemento creado y añadido a la lista.

[>> Página principal](#)

Crear y eliminar elementos dinámicamente

[Crear nuevo elemento](#) [Eliminar último elemento](#)

- Elemento inicial 1
- Elemento inicial 2
- Nuevo elemento creado

Último elemento eliminado.

[>> Página principal](#)



HTML5 - CSS3 - JS6 – DOM



Bloque 6 – addEventListener() y removeEventListener()

Bloque 6 – addEventListener() y removeEventListener()



Bloque 6 – addEventListener() y removeEventListener()

Objetivo. Aprender a activar y desactivar eventos sobre un elemento del DOM. En este bloque enseñaremos cómo usar removeEventListener() correctamente, comprendiendo la necesidad de utilizar funciones con nombre en lugar de funciones anónimas.

Paso 1 – Preparar el HTML

```
<main>
  <div class="container">
    <div class="card">
      <div class="cardTitle">Activar y desactivar eventos con addEventListener() y removeEventListener()</div>
      <div class="CardBody">
        <button id="activar"> Activar color de fondo</button>
        <button id="desactivar"> Desactivar evento</button>

        <p id="zona">Pasa el ratón por encima de este párrafo para cambiar su color de fondo.</p>

        <div id="resultado"></div>
      </div>
      <div class="cardFooter">
        <a href="../../index.html">&gt;&gt; Página principal</a>
      </div>
    </div>
  </div>
</main>
```



Bloque 6 – addEventListener() y removeEventListener()

Paso 2 – Preparar el JS

ejer06/js/script.js

```
"use strict"; // Siempre activamos el modo estricto

// 1. Capturamos los elementos
const activar = document.getElementById("activar");
const desactivar = document.getElementById("desactivar");
const zona = document.getElementById("zona");
const resultado = document.getElementById("resultado");

// 2. Declaramos la función que queremos ejecutar al pasar el
ratón
function cambiarFondo() {
  zona.style.backgroundColor = "lightblue";
  resultado.innerHTML = "<p>✓ Evento ejecutado: color de
fondo aplicado.</p>";
}
```

```
// 3. Activar el evento (lo añadimos al párrafo con
addEventListener)
activar.addEventListener("click", () => {
  zona.addEventListener("mouseover", cambiarFondo);
  resultado.innerHTML = "<p>● Evento activado. Pasa el ratón
por el párrafo.</p>";
});

// 4. Desactivar el evento (usamos removeEventListener)
desactivar.addEventListener("click", () => {
  zona.removeEventListener("mouseover", cambiarFondo);
  zona.style.backgroundColor = ""; // Quitamos el color
  resultado.innerHTML = "<p>● Evento desactivado. Ya no se
aplicará el color.</p>";
});
```



Bloque 6 – addEventListener() y removeEventListener()

Paso 3 – Explicación didáctica para tus alumnos

Método	¿Para qué sirve?
addEventListener()	Añade un evento a un elemento. Permite usar varias funciones para el mismo evento.
removeEventListener()	Elimina un evento previamente añadido. Solo funciona si la función es la misma y tiene nombre .

Regla importante:

Si usas una función anónima, como esta:

```
zona.addEventListener("mouseover", function () {  
    // código  
});
```

No podrás quitar ese evento después, porque removeEventListener() no sabrá a qué función te refieres.

Por eso siempre usamos una función con nombre (como cambiarFondo) si queremos desactivarla más tarde.



Bloque 6 – addEventListener() y removeEventListener()

Paso 4 – Resultado esperado

1. Al pulsar “🎯 Activar color de fondo”, se activa un evento mouseover sobre el párrafo.
2. Al pasar el ratón por encima del párrafo, el fondo cambia a azul claro.
3. Al pulsar “🔴 Desactivar evento”, se elimina el evento y ya no cambia el fondo.

Activar y desactivar eventos con addEventListener() y removeEventListener()

Pasa el ratón por encima de este párrafo para cambiar su color de fondo.

Activar y desactivar eventos con addEventListener() y removeEventListener()

Pasa el ratón por encima de este párrafo para cambiar su color de fondo.

Evento ejecutado: color de fondo aplicado.

[>> Página principal](#)

Activar y desactivar eventos con addEventListener() y removeEventListener()

Pasa el ratón por encima de este párrafo para cambiar su color de fondo.

Evento desactivado. Ya no se aplicará el color.

[>> Página principal](#)



HTML5 - CSS3 - JS6 – DOM



Bloque 7 – Navegación por el árbol del DOM

Bloque 7 – Navegación por el árbol del DOM



Bloque 7 – Navegación por el árbol del DOM

(Métodos: parentNode, childNodes, firstChild, lastChild, nextSibling, previousSibling)

Objetivo. Aprender a navegar entre nodos del DOM sin usar selectores. En este bloque veremos cómo movernos desde un nodo a su padre, hijos o hermanos usando propiedades especiales del DOM.

Paso 1 – Preparar el HTML

ejer07/index.html

```
<main>
  <div class="container">
    <div class="card">
      <div class="cardTitle">Navegación entre nodos del DOM</div>
      <div class="CardBody">
        <ul id="listaNombres">
          <li>Elemento 1</li>
          <li id="elementoActual">Elemento 2 (actual)</li>
          <li>Elemento 3</li>
        </ul>
      </div>
    </div>
  </div>
</main>
```

```
<button id="verPadre">Mostrar nodo padre</button>
  <button id="verHijos">Mostrar hijos del padre</button>
  <button id="verAnterior">Mostrar hermano
anterior</button>
  <button id="verSiguiente">Mostrar hermano
siguiente</button>

  <div id="resultado"></div>
  </div>
  <div class="cardFooter">
    <a href="../../index.html">&gt;&gt; Página principal</a>
  </div>
</div>
</main>
```



Bloque 7 – Navegación por el árbol del DOM

Paso 2 – Preparar el JS

ejer07/js/script.js

```
"use strict"; // Siempre activado

// Capturas
const actual = document.getElementById("elementoActual");
const verPadre = document.getElementById("verPadre");
const verHijos = document.getElementById("verHijos");
const verAnterior = document.getElementById("verAnterior");
const verSiguiente = document.getElementById("verSiguiente");
const resultado = document.getElementById("resultado");

// Mostrar el nodo padre
verPadre.addEventListener("click", () => {
  const padre = actual.parentNode;
  resultado.innerHTML = `<p>█ Nodo padre:  
<code>${padre.nodeName}</code> con ID: "${padre.id}"</p>`;
});

// Mostrar todos los hijos del padre
verHijos.addEventListener("click", () => {
  const hijos = actual.parentNode.childNodes;
  let lista = "<p>█ Hijos del padre (incluye nodos de texto):</p><ul>";
  hijos.forEach((hijo) => {
    lista += `<li><code>${hijo.nodeName}</code> (tipo:  
${hijo.nodeType})</li>`;
  });
  lista += "</ul>";
  resultado.innerHTML = lista;
});
```

```
// Mostrar hermano anterior
verAnterior.addEventListener("click", () => {
  const anterior = actual.previousSibling;
  if (anterior && anterior.nodeType === 1) {
    resultado.innerHTML = `<p>█ Hermano anterior:  
<code>${anterior.textContent}</code></p>`;
  } else {
    resultado.innerHTML = `<p>⚠ No hay hermano anterior o es un nodo de  
texto.</p>`;
  }
});

// Mostrar hermano siguiente
verSiguiente.addEventListener("click", () => {
  const siguiente = actual.nextSibling;
  if (siguiente && siguiente.nodeType === 1) {
    resultado.innerHTML = `<p>█ Hermano siguiente:  
<code>${siguiente.textContent}</code></p>`;
  } else {
    resultado.innerHTML = `<p>⚠ No hay hermano siguiente o es un nodo  
de texto.</p>`;
  }
});
```



Bloque 7 – Navegación por el árbol del DOM

Paso 3 – Explicación

Propiedad	¿Qué devuelve?
parentNode	El nodo padre del actual
childNodes	Lista de todos los nodos hijos , incluidos nodos de texto (espacios, saltos)
firstChild	El primer nodo hijo (incluso si es un espacio o salto de línea)
lastChild	El último nodo hijo
nextSibling	El nodo hermano siguiente
previousSibling	El nodo hermano anterior

Importante:

- A veces obtendréis nodos tipo `#text`, que son espacios o saltos de línea.
- Para aseguraros de que un nodo es un elemento HTML real, comprobad que su `nodeType` es 1.

```
if (nodo.nodeType === 1) { ... } // solo elementos HTML
```



Bloque 7 – Navegación por el árbol del DOM

Nota: Visualmente parece que es seguido de otro , pero internamente, el navegador incluye también los saltos de línea y espacios como nodos de texto (#text).

Visualización interna del DOM	
<pre><ul id="listaNombres"> Elemento 1 <li id="elementoActual">Elemento 2 (actual) Elemento 3 </pre>	<pre> #text Elemento 1 #text <li id="elementoActual">Elemento 2 (actual) #text Elemento 3 #text </pre>

¿Qué aprendemos de esto?

🎓 Cuando queráis moveros por el árbol del DOM, es importante distinguir entre:

- ...Sibling: incluye también nodos de texto invisibles (saltos de línea).
- ...ElementSibling: solo se mueve entre etiquetas HTML (nodos tipo 1).



Bloque 7 – Navegación por el árbol del DOM

Paso 4 – Resultado esperado

1. Al pulsar “Mostrar nodo padre”, se muestra el nombre del nodo padre (UL) y su ID.
2. Al pulsar “Mostrar hijos del padre”, se listan todos los hijos del nodo UL (pueden aparecer #text).
3. Al pulsar “Mostrar hermano anterior”, se muestra el contenido del anterior (si es un nodo válido).
4. Al pulsar “Mostrar hermano siguiente”, se muestra el contenido del siguiente (si es un nodo válido).



Ejercicios

HTML5 - CSS3 - JS6 – DOM



Ejercicios



HTML5 - CSS3 - JS6 – DOM



Ejercicios. Enunciado



Ejercicio

Implementa los ejemplos anteriores en tu propia plantilla.