

Elemento / Método / Propiedad	Ejemplo	Descripción
<code>document.getElementById(id)</code>	<code>const btn = document.getElementById("evento");</code>	Devuelve el primer (y único) elemento con el id especificado.
<code>document.getElementsByClassName(nombre)</code>	<code>const cajas = document.getElementsByClassName("caja");</code>	Devuelve una <code>HTMLCollection</code> viva de elementos con la clase dada.
<code>document.getElementsByTagName(nombre)</code>	<code>const divs = document.getElementsByTagName("div");</code>	Devuelve una <code>HTMLCollection</code> viva de elementos con la etiqueta dada.
<code>document.querySelector(selector)</code>	<code>const primerBoton = document.querySelector(".opcion");</code>	Devuelve el primer elemento que coincide con el selector CSS.
<code>document.querySelectorAll(selector)</code>	<code>const botones = document.querySelectorAll(".opcion");</code>	Devuelve un <code>NodeList</code> estático con todos los elementos que coinciden con el selector.
<code>element.style.propiedad</code>	<code>boton.style.backgroundColor = "blue";</code>	Permite leer o escribir estilos en línea directamente.
<code>element.classList</code>	<code>menu.classList.toggle("show");</code>	Objeto que permite manipular las clases de un elemento.
<code>element.classList.add(nombre)</code>	<code>boton.classList.add("active");</code>	Añade una clase al elemento.
<code>element.classList.remove(nombre)</code>	<code>boton.classList.remove("active");</code>	Elimina una clase del elemento.
<code>element.classList.toggle(nombre)</code>	<code>document.body.classList.toggle("dark-mode");</code>	Añade la clase si no está; la elimina si está.
<code>element.classList.contains(nombre)</code>	<code>if (boton.classList.contains("active")) { ... }</code>	Devuelve <code>true</code> si el elemento tiene la clase, <code>false</code> si no.
<code>document.createElement(tag)</code>	<code>const p = document.createElement("p");</code>	Crea un nuevo nodo de elemento (no lo inserta aún en el DOM).
<code>document.createTextNode(texto)</code>	<code>const texto = document.createTextNode("Hola");</code>	Crea un nodo de texto.

<code>padre.appendChild(hijo)</code>	<code>contenedor.appendChild(p);</code>	Añade un nodo como último hijo del elemento padre.
<code>padre.insertBefore(nuevo, existente)</code>	<code>contenedor.insertBefore(h2, titulo.nextSibling);</code>	Inserta un nodo antes de un nodo hermano existente.
<code>padre.replaceChild(nuevo, viejo)</code>	<code>contenedor.replaceChild(nuevoH1, viejoH1);</code>	Reemplaza un nodo hijo por otro.
<code>padre.removeChild(hijo)</code>	<code>contenedor.removeChild(ultimoParraro);</code>	Elimina un nodo hijo del padre.
<code>element.innerHTML</code>	<code>resultado.innerHTML = "Contador: " + contador;</code>	Obtiene o establece el contenido HTML de un elemento.
<code>element.textContent</code>	<code>mensaje.textContent = "Palabra: " + palabra;</code>	Obtiene o establece el texto plano (sin interpretar HTML).
<code>element.value</code>	<code>let letra = input.value.toLowerCase();</code>	Obtiene o establece el valor de un campo de formulario.
<code>addEventListener(evento, función)</code>	<code>boton.addEventListener("click", cambiarColor);</code>	Asocia una función a un evento sin sobrescribir otros listeners.
<code>removeEventListener(evento, función)</code>	<code>boton.removeEventListener("click", cambiarColor);</code>	Elimina un listener previamente añadido (misma referencia de función).
<code>event.target</code>	<code>if (event.target.tagName === "BUTTON") { ... }</code>	Elemento que disparó el evento (clave para delegación).
<code>event.preventDefault()</code>	<code>e.preventDefault(); // en contextmenu</code>	Cancela la acción predeterminada (ej: menú contextual).
<code>window.location.reload()</code>	<code>location.reload();</code>	Recarga la página actual.
<code>window.open(url)</code>	<code>window.open("http://www.google.es");</code>	Abre una nueva pestaña/ventana con la URL dada.
<code>window.alert(mensaje)</code>	<code>alert("Fin del tiempo");</code>	Muestra un cuadro de diálogo de alerta.
<code>window.prompt(mensaje)</code>	<code>let nombre = prompt("¿ Tu nombre?");</code>	Muestra un cuadro para que el usuario ingrese texto.

<code>setInterval(función, ms)</code>	<code>intervalo = setInterval(actualizar, 1000);</code>	Ejecuta una función repetidamente cada ms milisegundos.
<code>setTimeout(función, ms)</code>	<code>setTimeout(() => alert("Listo"), 2000);</code>	Ejecuta una función una sola vez tras ms milisegundos.
<code>clearInterval(id)</code>	<code>clearInterval(intervalo);</code>	Detiene un temporizador de <code>setInterval</code> .
<code>clearTimeout(id)</code>	<code>clearTimeout(temporizador);</code>	Detiene un temporizador de <code>setTimeout</code> .
<code>new Date()</code>	<code>const ahora = new Date();</code>	Crea un objeto con la fecha y hora actual.
<code>Date.getTime()</code>	<code>let dif = fin.getTime() - inicio.getTime();</code>	Devuelve milisegundos desde 1/1/1970. Útil para calcular diferencias.
<code>localStorage.setItem(clave, valor)</code>	<code>localStorage.setItem("color", "#ff0000");</code>	Guarda un string en <code>localStorage</code> (persistente).
<code>localStorage.getItem(clave)</code>	<code>const color = localStorage.getItem("color");</code>	Recupera un valor de <code>localStorage</code> por clave.
<code>localStorage.removeItem(clave)</code>	<code>localStorage.removeItem("usuario");</code>	Elimina un ítem de <code>localStorage</code> .
<code>sessionStorage</code>	<code>sessionStorage.setItem("temp", "dato");</code>	Igual que <code>localStorage</code> , pero solo dura la sesión (pestaña).
<code>document.cookie</code>	<code>document.cookie = "nombre=Ana; expires=...";</code>	Cadena para crear/borrar cookies (formato especial).
<code>encodeURIComponent(valor)</code>	<code>encodeURIComponent(nombre)</code>	Codifica un valor para usarlo en cookies o URLs.
<code>decodeURIComponent(valor)</code>	<code>decodeURIComponent(partes[0])</code>	Decodifica un valor previamente codificado.
<code>JSON.stringify(objeto)</code>	<code>localStorage.setItem("datos", JSON.stringify(usuario));</code>	Convierte un objeto/array en una cadena JSON.
<code>JSON.parse(cadena)</code>	<code>const usuario = JSON.parse(localStorage.getItem("datos"));</code>	Convierte una cadena JSON en un objeto/array.

<code>window.addEventListener("storage", ...)</code>	<code>window.addEventListener("storage", (e) => { ... });</code>	Se dispara en otras pestañas cuando cambia el almacenamiento.
<code>element.tagName</code>	<code>if (event.target.tagName === "BUTTON")</code>	Devuelve el nombre de la etiqueta en mayúsculas (ej: "BUTTON").
<code>Array.prototype.push()</code>	<code>notas.push({asignatura, nota, profesor});</code>	Añade un elemento al final del array.
<code>Array.prototype.splice(índice, cuántos)</code>	<code>notas.splice(pos, 1);</code>	Elimina (o añade) elementos en una posición dada.
<code>Array.prototype.sort(comparador)</code>	<code>notas.sort((a, b) => b.nota - a.nota);</code>	Ordena el array (numérico descendente en este caso).
<code>Array.prototype.forEach(función)</code>	<code>botones.forEach(btn => btn.classList.remove("active"));</code>	Ejecuta una función por cada elemento del array.
<code>String.prototype.padStart(longitud, relleno)</code>	<code>String(min).padStart(2, '0') → "05"</code>	Rellena una cadena a la izquierda hasta alcanzar la longitud dada.
<code>Object.keys(objeto)</code>	<code>const usuarios = Object.keys(localStorage);</code>	Devuelve un array con las claves de un objeto.