

# Actividades unidad 4: Atributos y propiedades, Clases y estilos, Manipulación del DOM y Eventos



## 1. Atributos y Propiedades

Dado el **Ej1.html** programa el código JavaScript necesario para manipular atributos y propiedades del DOM según las acciones del usuario:

1. **btnVer:** mostrar en consola los atributos src de la imagen y href del enlace, y las propiedades width y target.
2. **btnCambiar:** actualizar src a "img2.jpg" y href a "https://www.youtube.com".
3. **btnQuitar:** eliminar alt de la imagen y target del enlace.
4. **btnComprobar:** alertar si la imagen tiene alt y si el enlace tiene target.
5. Mostrar en consola la diferencia entre atributo y propiedad de la imagen (`getAttribute("width")` vs `img.width`).

## 2. Manipulación de clases y estilos

Dado el **Ej2.html** gestiona las clases y estilos de un mensaje según las acciones del usuario:

1. **btnAdd:** añadir clases resaltado y alerta al mensaje.
2. **btnRemove:** quitar resaltado y alerta.
3. **btnToggle:** alternar solo la clase resaltado.
4. **btnStyle:** cambiar en línea `fontStyle` a italic y border a 2px solid red.
5. Mostrar en consola si el elemento contiene la clase alerta usando `contains`.

### 3. Manipulación de contenido y estructura del DOM

Dado el **Ej3.html** implementa el código JavaScript necesario para modificar el contenido y la estructura del DOM a partir de la interacción del usuario:

1. **btnCambiarTexto**: modificar el primer párrafo con `innerText` y el segundo con `innerHTML` incluyendo un `<strong>`.
2. **btnAñadirElemento**: crear un nuevo `<p>` con texto "Párrafo añadido" y añadirlo al final.
3. **btnReemplazar**: reemplazar el primer párrafo por uno nuevo con "Primer párrafo reemplazado".
4. **btnEliminar**: eliminar el último párrafo de `#contenido`.
5. Clonar el primer párrafo y añadirlo al final usando `cloneNode(true)`.

### 4. Eventos y `addEventListener`

Dado el **Ej4.html** gestiona los eventos de un menú utilizando delegación de eventos y las propiedades del objeto `event`:

1. **Asociar un click** al contenedor `#menu` (delegación).
2. **Cambiar el fondo** del botón pulsado (`event.target`) a verde y el **borde** del contenedor (`event.currentTarget`) a negro.
3. **Mostrar** en `#info` el **texto** del botón pulsado.
4. Evitar que el borde del contenedor se aplique si se hace clic fuera de un botón.

### 5. Delegación de eventos con `matches()` y `closest()`

Dado el **Ej5.html** implementa un sistema de delegación de eventos que permita interactuar con tarjetas existentes y con otras que se creen dinámicamente:

1. Usar un único `addEventListener` en `#tarjetas`.
2. Comprobar con `matches` que **solo actúe** si se **pulsa** un **botón**.
3. **Resaltar** la **tarjeta** del botón pulsado (`closest`) cambiando su **fondo** a **amarillo**.
4. **Añadir** un **botón dinámico** que **cree nuevas tarjetas** y ver que la delegación funciona también para ellas.

## 6. Eventos de carga y asignación directa

Dado el **Ej6.html** Inicializa correctamente el código JavaScript y gestiona eventos de diferentes formas sin provocar errores de carga:

1. Usar **DOMContentLoaded** para **cambiar** el **textContent** del título a "DOM listo para JS".
2. **Asignar** directamente a **myButton.onclick** una función que muestre un **alert "Botón clickeado"**.
3. **Añadir y eliminar** el mismo listener con **addEventListener** y **removeEventListener** para ver cómo se comporta.