Resumen acceso DOM con JavaScript

Tabla de contenido

Acceder a una etiqueta HTML por su id 1
Acceder a unas etiquetas HTML por su etiqueta 2
Acceder a unas etiquetas HTML por su clase 2
Acceder a una etiqueta HTML usando un selector CSS 3
Acceder a varias etiquetas HTML usando un selector CSS 3
No partir de document 3
Añadir un evento a un elemento 3
Añadir un evento a muchos elementos 4
Crear un elemento 4
Obtener, añadir o modificar el texto de un elemento "normal" 4
Borrar todo el contenido de un elemento "normal" 5
Obtener, añadir o modificar el texto de un elemento de formulario 5
Añadir o modificar un estilo de un elemento 5
Añadir una clase a un elemento 5
Eliminar una clase a un elemento 6
Intercambiar una clase a un elemento 6
Eliminar un elemento 6
Acceder al padre o abuelo de un elemento 6
Acceder a todos los hijos de un elemento 6
Acceder al siguiente hermano de un elemento 7
Acceder al hermano anterior de un elemento 7
Añadir o acceder a atributos de un elemento 7
Añadir o acceder a atributos propios data- de un elemento 7
Recorrer un array de objetos JSON 7
Recorrer un array con un array de objetos JSON 8
Usar target y currentTarget en un evento 8

Acceder a una etiqueta HTML por su id

Usaremos:

```
document.getElementById("nombreId")
```

Esto devuelve el nodo con lo que debemos guardarlo en una variable o usarlo en la misma línea.

Ejemplo:

```
// La variable / constante no tiene por qué llamarse igual al id
const contenido = document.getElementById("contenido");
```

Ejemplo:

```
document.getElementById("contenido").textContent = "Hola";
```

Posible fallo

Si se muestra un error en la consola del navegador que ponga algo como lo siguiente:

```
Uncaught TypeError: document.getElementById(...) is null
```

Puede ser debido a los siguientes fallos:

- hemos puesto el id mal
- no hemos añadido defer en la etiqueta script
- hemos puesto un archivo js diferente en la etiqueta script

Acceder a unas etiquetas HTML por su etiqueta

Usaremos:

```
document.getElementsByTagName("nombreEtiquetaHTML")
```

Esto devuelve una colección con todos los elementos que sean de esa etiqueta (una HTMLCollectionOf) con lo que lo normal será usar un for o for of para recorrerlos. Si no hay ninguna etiqueta de ese tipo, devuelve una colección vacía ([]).

Ejemplo:

```
const parrafos = document.getElementsByTagName("p");
for(const p of parrafos) {
    // Aquí haremos algo con el párrafo
    // ...
}
```

Obtener solo la primera etiqueta

Si solo nos interesa la primera etiqueta, podemos usar lo siguiente:

```
const parrafo = document.getElementsByTagName("p")[0];
```

Acceder a unas etiquetas HTML por su clase

Usaremos:

```
document.getElementsByClassName("nombreClaseCSS")
```

Esto devuelve una colección con todos los elementos que tenga al menos esa clase, pudiendo tener otras (una HTMLCollectionOf) con lo que lo normal será usar un for o for of para recorrerlos. Si no hay ninguna etiqueta de ese tipo, devuelve una colección vacía ([]).

Ejemplo:

```
const mensajes = document.getElementsByClassName("mensaje");
for(const m of mensajes) {
    // Aquí haremos algo con el elemento
    // ...
}
```

Obtener solo el primero

Si solo nos interesa la primera etiqueta, podemos usar lo siguiente:

```
const mensaje = document.getElementsByClassName("mensaje")[0];
```

Acceder a una etiqueta HTML usando un selector CSS

Usaremos:

```
document.querySelector("selector")
```

Esto devuelve el primero elemento que cumplan el selector indicado, pudiendo ser cualquier selector de CSS . Si no hay ninguno devuelve null

Ejemplo:

```
// Accedemos al primer p dentro del primer div
const elemento = document.querySelector("div p");
```

Acceder a varias etiquetas HTML usando un selector CSS

Usaremos:

```
document.querySelectorAll("selector")
```

Esto devuelve una colección ((una NodeListOf) con todos los elementos que cumplan el selector indicado con lo que lo normal será usar un for o for of para recorrerlos. Si no hay ningún elemento que lo cumpla, devuelve una colección vacía ([]).

Ejemplo:

```
const parrafos = document.querySelectorAll("div p");
for(const p of parrafos) {
    // Aquí haremos algo con el elemento
    // ...
}
```

Obtener solo la primera etiqueta

Si solo nos interesa la primera etiqueta, podemos usar lo siguiente:

```
const parrafo = document.querySelectorAll("div p")[0];
```

No partir de document

No es obligatorio usar siempre document., pudiendo usar cualquier otro elemento a partir del que buscar.

Ejemplo:

```
const contenido = document.getElementById("contenido");
const mensajes = contenido.getElementsByClassName("mensaje");
```

Ejemplo:

```
// Si no necesitamos el contenido podemos usar
const mensajes = document.getElementById("contenido").getElementsByClassName("mensaje");
```

Añadir un evento a un elemento

Usaremos cualquiera de los métodos anteriores, getElementById, getElementsByClassName, getElementsByTagName, querySelector o querySelectorAll y usar el método **addEventListener**.

Uso:

```
.addEventListener("evento", función_a_ejecutar_cuando_se_produzca_el_evento)
```

Ejemplo:

```
document.getElementById("contenido").addEventListener("click", mostrar);
```

Podemos añadir todos los eventos que queramos a un mismo elemento e incluso el mismo evento más de una vez.

Añadir un evento a muchos elementos

Podemos recorrer una colección de elementos para asignarles el mismo evento.

Ejemplo:

```
// Cogemos todas las etiquetas p
const párrafos = document.getElementsByTagName("p");
for(p of párrafos) {
    p.addEventListener("click", cambiar); // Suponemos que existe una función llamada cambiar
}
```

Crear un elemento

Usaremos:

```
document.createElement("etiqueta")
```

Este método devuelve el elemento creado con lo que lo normal será guardarlo en una variable o constante.

Ejemplo:

```
const div = document.createElement("div");
```

Luego casi siempre deberemos añadirlo a otro elemento existente o a document.body mediante el método **appenChild**, **insertBefore**, ...

Ejemplo:

```
// Suponemos que existe un elemento con id contenedor
const contenedor = document.getElementById("contenedor");
const p = document.createElement("p");
// Haremos algo con el p (añadirle texto, estilos, clases, otros elementos, ...)
// ...
contenedor.appendChild(p);
```

Ejemplo

```
// Suponemos que existe un elemento con id contenedor
const contenedor = document.getElementById("contenedor");
const ul = document.createElement("ul"); // Creamos un ul
const li1 = document.createElement("li"); // Creamos un li
li1.textContent = "Lugo"; // Le añadimos texto
ul.appendChild(li1); // Añadimos el li al ul
const li2 = document.createElement("li"); // Creamos otro li
li2.textContent = "Ponecetra"; // Le añadimos texto
ul.appendChild(li2); // Añadimos el li al ul
contenedor.appendChild(ul); // Añadimos el ul al contenedor
```

Obtener, añadir o modificar el texto de un elemento "normal"

Basta usar la propiedad **textContent** del mismo.

Ejemplo:

```
// Suponemos que existe un elemento con id contenedor
const contenedor = document.getElementById("contenedor");
contenedor.textContent = "Nuevo texto";
```

También podemos usar **innertHTML** si lo que queremos es asignarle alguna etiqueta HTML y no tener que andar creando uno o varios elementos.

Ejemplo:

```
// Suponemos que existe un elemento con id contenedor
const contenedor = document.getElementById("contenedor");
contenedor.innerHTML = "<div class='mensaje'><p>Hola</p></div>";
```

Borrar todo el contenido de un elemento "normal"

Basta con asignar la cadena vacía a la propiedad **textContent** del elemento deseado.

Ejemplo:

```
// Suponemos que existe un elemento con id contenedor
const contenedor = document.getElementById("contenedor");
contenedor.textContent = "";
```

Obtener, añadir o modificar el texto de un elemento de formulario

En la mayoría de los elementos usaremos la propiedad **value** del mismo.

Ejemplo:

```
// Suponemos que existe un elemento input con id nombre
const nombre = document.getElementById("nombre").value;
```

Añadir o modificar un estilo de un elemento

Ya sea un nuevo elemento como uno existente bastará poner:

```
elemento.style.propiedad = "valor";
```

Ejemplo:

```
const contenedor = document.getElementById("contenedor");
contenedor.style.backgroundColor = "red";
contenedor.style.fontSize = "2rem";
```

Si el valor es un número no hace falta poner comillas.

Si el valor es una variable, un .value, ... podemos hacer:

```
const valor = document.getElementById("numero").value;
const contenedor = document.getElementById("contenedor");
contenedor.style.fontSize = `${valor}rem`;
// O
contenedor.style.fontSize = valor + "rem";
```

Añadir una clase a un elemento

Ya sea un nuevo elemento como uno existente bastará poner:

```
elemento.classList.add("nombreClase");

0

elemento.classList.add("nombreClase1", "nombreClase2", ...);
```

Ejemplo:

```
const contenedor = document.getElementById("contenedor");
// Suponemos que tenemos una clase llamada mensaje y otra caja
contenedor.classList.add("mensaje", "caja");
```

Eliminar una clase a un elemento

Ya sea un nuevo elemento como uno existente bastará poner:

```
elemento.classList.remove("nombreClase");

0

elemento.classList.remove("nombreClase1", "nombreClase2", ...);
```

Ejemplo:

```
const contenedor = document.getElementById("contenedor");
// Suponemos que tenemos una clase llamada mensaje y otra caja
contenedor.classList.remove("mensaje", "caja");
```

Intercambiar una clase a un elemento

Usando el método **toggle** podemos hacer que una clase se añada a un elemento si no la tiene o se quite si la tiene.

Ejemplo:

```
const contenedor = document.getElementById("contenedor");
// Suponemos que tenemos una clase llamada mensaje
contenedor.classList.toggle("mensaje");
```

Eliminar un elemento

Basta con usar el método **remove** del elemento deseado.

Ejemplo:

```
const contenedor = document.getElementById("contenedor");
contenedor.remove();
```

Acceder al padre o abuelo de un elemento

Si necesitamos acceder al padre de un elemento usaremos su propiedad **parentNode**. Si necesitamos acceder al abuelo de un elemento usaremos **parentNode.parentNode** y así sucesivamente.

Ejemplo:

```
const texto = document.getElementById("texto");
// Cambiamos el color de fondo de su padre
texto.parentNode.style.backgroundColor = "red";
```

Acceder a todos los hijos de un elemento

Usaremos la propiedad **childNodes** o **children** del elemento deseado. La diferencia es que el primero también de vuelve los nodos de texto (como un Enter entre etiquetas) y comentarios, con lo que suele ser mejor usar el segundo.

Ambos devuelven una colección con los hijos que normalmente recorreremos con un for o for of. Si no tiene hijos devuelven una colección vacía [].

Ejemplo:

```
const contenedor = document.getElementById("contenedor");
// Ponemos a rojo todos los hijos
for(const hijo of contenedor.children) {
    hijo.style.color = "red";
}
```

Acceder al siguiente hermano de un elemento

Usaremos la propiedad **nextSibling** o **nextElementSibling** del elemento deseado. La diferencia es que el primero también de vuelve los nodos de texto (como un Enter entre etiquetas) y comentarios, con lo que suele ser mejor usar el segundo. Si no lo hay devuelve null.

Ejemplo:

```
const nombre = document.getElementById("nombre");
// Ponemos a rojo su primer hermano siguiente
nombre. nextElementSibling.style.color = "red";
```

Acceder al hermano anterior de un elemento

Usaremos la propiedad **previousSibling** o **previousElementSibling** del elemento deseado. La diferencia es que el primero también de vuelve los nodos de texto (como un Enter entre etiquetas) y comentarios, con lo que suele ser mejor usar el segundo. Si no lo hay devuelve null.

Añadir o acceder a atributos de un elemento

Accederemos directamente al atributo deseado del elemento.

Ejemplo:

```
const imagen = document.getElementById("imagen");
imagen.src = "casa.png";
image.alt = "Imagen de una casa";
```

Añadir o acceder a atributos propios data- de un elemento

Los atributos propios que creemos se recomiendan que se llamen data-nombreDeseado para que no hayan conflicto con atributo HTML y sean fáciles de distinguir.

Para añadir o acceder a un atributo data- usaremos la propiedad dataset.nombreAtributo del elemento deseado.

Ejemplo:

```
// Suponemos que tenemos un <div id="notas" data-numero="3>
const notas = document.getElementById("notas");
console.log(notas.dataset.numero);
// Añadimos un nuevo atributo data
notas.dataset.estado = "valido";
```

Recorrer un array de objetos JSON

Usaremos un for o un for f.

Ejemplo:

```
const personas = [
    {"nombre": "Laura", edad: 35}, {"nombre": "segio", edad: 41}, {"nombre": "eva", edad: 21}];
for(const p of personas) {
    console.log(p.nombre);
}
```

Recorrer un array con un array de objetos JSON

Usaremos un for o un for of y luego otro for o for of

Ejemplo:

```
const personas = [
    {"nombre": "Laura", "edad": 35,
        "notas": [
           {"asignatura": "Matemáticas", "nota": 8.5},
           {"asignatura": "Historia", "nota": 9.0}
    ]}
    , {"nombre": "Sergio", "edad": 41,
           "notas": [
                  {"asignatura": "Física", "nota": 7.5},
                  {"asignatura": "Química", "nota": 8.0}
    ]}
    , {"nombre": "Eva", "edad": 21,
           "notas": [
                  {"asignatura": "Literatura", "nota": 9.5},
                   {"asignatura": "Arte", "nota": 10.0}
    ]}
];

for(const p of personas) {
    console.log(p.nombre, p.edad);
    for(const materia of p.notas) {
           console.log(materia.asignatura, materia.nota);
    }
}
```

Usar target y currentTarget en un evento

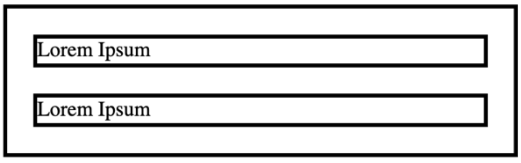
Si usamos la misma función para varios elementos de un evento, podemos usar una de las siguientes propiedades para reconocer el elemento:

- **target**: contiene siempre el elemento donde se produjo el evento realmente, se haya suscrito o no al evento
- **currentTarget**: contiene el elemento actual de la fase de burbujeo que se suscribió al evento

¿Qué es eso del burbujeo?
Cuando se produce un evento en un elemento, si está suscrito al mismo, se ejecuta la función indicada. Luego se comprueba si su padre está también suscrito y se ejecutará la función indicada si es así, luego en el abuelo, ...

Ejemplo:

```
<div id="uno">
        <div id="dos">Lorem Ipsum</div>
        <div id="tres">Lorem Ipsum</div>
    </div>
    <script>
        function mostrar(evt) {
```



```
            console.log("target: " + evt.target.id, "currentTarget: " + evt.currentTarget.id);
        }
        document.getElementById("uno").addEventListener("click", mostrar);
        document.getElementById("dos").addEventListener("click", mostrar);
        document.getElementById("uno").addEventListener("click", mostrar);
    </script>
```

Cuando se haga clic en el div uno pero no dentro del div dos o tres:

```
target: uno currentTarget: uno
```

Cuando se haga clic en el div dos:

```
target: dos currentTarget: dos
target: dos currentTarget: uno
```

Cuando se haga clic en el div tres:

```
target: tres currentTarget: uno
```

Resumen

Si el padre no está suscrito al mismo evento, nos da igual target o currentTarget, si no debemos usar target si queremos saber el elemento exacto donde se produjo el evento.