**M1S8: DOM (Selección y Manipulación desde el Navegador) y Eventos**

**¡Hola! ya te encuentras en la semana 8 de tu U Camp, ¿Cómo te sientes? ¿Has aprendido mucho? En esta semana tienes que empezar a diseñar y desarrollar tu segundo proyecto para que lo puedas mostrar en la siguiente U Class, recuerda que si presentas alguna duda, puedes consultar con tus coaches.**

**Te recuerdo que ya aprendiste sobre: HTML, CSS, Flexbox, JavaScript y por supuesto ya entregaste tu primer proyecto; a continuación, iniciamos con el contenido de DOM. ¡Sigue adelante, nos vemos!**

**ÍNDICE**

* [DOM: Document Object Model](https://github.com/U-Camp/BOOT-M1-SEM8#dom-document-object-model)
  + [Selección](https://github.com/U-Camp/BOOT-M1-SEM8#selecci%C3%B3n)
  + [Manipulación](https://github.com/U-Camp/BOOT-M1-SEM8#manipulaci%C3%B3n)
* [Eventos](https://github.com/U-Camp/BOOT-M1-SEM8#eventos)

**DOM: Document Object Model**

Ahora bien, ¿Qué pasa cuando el navegador web carga el código HTML?

Entra en función el intérprete de los navegadores (render engine), el cual conforme va leyendo el código va generando en memoria un modelo con los elementos HTML interpretados al cual se le llama DOM.

Cuando un navegador web carga cualquier página de internet, ésta crea un árbol enlazando cada elemento padre con sus hijos así con los atributos y estilos asociados a cada uno de ellos.

A este árbol de componentes se le llama **DOM (Document Object Model)**. Este árbol de componentes es importante, ya que gracias a él se puede acceder a los elementos de la página y modificarlos para agregarles funcionalidad y que la web deje de ser estática para pasar a ser dinámica; esto se realiza usando JavaScript.

Para poder acceder a través del DOM a nuestro documento HTML, es necesario comprender dos partes:

1. Selección
2. Manipulación

Estas dos acciones nos permitirán obtener cada elemento HTML como si fueran objetos y trabajar sus propiedades, métodos y eventos.

El DOM es un estándar que nos permite obtener, cambiar, agregar o borrar elementos HTML, a través de JavaScript.

**Selección**

El primer paso para trabajar con DOM es seleccionar el elemento que buscamos.

Existen 5 formas principales para llegar a un elemento específico y asignarlo a una variable.

| **Método (Función)** | **Buscamos por...** | **Argumento del Método** |
| --- | --- | --- |
| getElementById() | ID | ID |
| getElementsByClassName() | Clase (CSS) | CLASE |
| getElementsByTagName() | Etiqueta HTML | ETIQUETA |
| querySelector() | Selector de primera coincidencia | ID O CLASE |
| querySelectorAll() | Selector de múltiples coincidencias | IDS O CLASES |

Detallemos cada una.

1. getElementById()

Se utiliza para seleccionar un elemento HTML utilizando el identificador (id) que se encuentra en su etiqueta.

Tomemos como ejemplo dos archivos, uno llamado index.html y otro index.js.

index.html

<div id="main">Esta es una etiqueta</div>

index.js

// Seleccionamos el elemento a través de su ID

const main = document.getElementById("main")

// Verificamos en consola que obtenemos la etiqueta

console.log(main)

1. getElementsByClassName

Se utiliza para seleccionar varios elementos a través de su clase establecida en class como atributo. De retorno, obtendremos una colección (array-like). Para acceder a cada elemento, debemos utilizar bracket notation. Posteriormente, puedes, a través de dot notation, acceder a una propiedad específica.

index.html

<div class="main">Esta es una etiqueta</div>

index.js

// Seleccionamos los elementos a través de su clase

const main = document.getElementsByClassName("main")

// Verificamos en consola que obtenemos la etiqueta

console.log(main[0])

console.log(main[0].textContent)

1. getElementsByTagName()

Muy similar al anterior, nos permite seleccionar elementos a través del nombre de las etiquetas. Buscarán todos los elementos que coincidan y los incluirán en una colección que podrán accederse a través de bracket notation y dot notation

index.html

<div class="main">Esta es una primera etiqueta</div>

<div class="main">Esta es una segunda etiqueta</div>

index.js

// Seleccionamos los elementos a través de su etiqueta

const main = document.getElementsByTagName("div")

// Verificamos en consola que obtenemos la etiqueta

console.log(main[0])

console.log(main[0].textContent)

console.log(main[1])

console.log(main[1].textContent)

1. querySelector

Este selector nos permite seleccionar, con la primera coincidencia, un elemento. Una de las ventajas de este selector es que puedes establecer una búsqueda por ID o por clase.

Es importante considerar que cuando queremos buscar por ID, coloquemos un numeral (#) y si es unan clase, un punto (.) dentro del argumento del método.

index.html

<div id="main">Esta es una primera etiqueta</div>

<div class="main">Esta es una segunda etiqueta</div>

index.js

// Seleccionamos el elemento a través de su ID o clase

const main = document.querySelector("#main")

// Verificamos en consola que obtenemos la etiqueta

console.log(main.textContent)

1. querySelectorAll

Muy similar a querySelector. Sin embargo, con este método encuentra todas las coincidencias de elementos y las integra a una colección.

index.html

<div id="main">Esta es una primera etiqueta</div>

<div class="main-content">Esta es una segunda etiqueta</div>

<div class="main-content">Esta es una tercera etiqueta</div>

index.js

// Seleccionamos los elementos a través de su ID o clase

const main = document.querySelectorAll(".main-content")

// Verificamos en consola que obtenemos la etiqueta

console.log(main)

console.log(main[0].textContent)

console.log(main[1].textContent)

Ahora bien, una vez seleccionados los elementos, es momento de manipularlos.

**Manipulación**

La manipulación puede establecerse, una vez seleccionado el elemento.

Podemos realizar modificaciones a través de sus atributos y contenido, usando diferentes métodos:

1. getAttribute: Retorna el valor de un atributo específico.

element.getAttribute('src')

1. setAttribute: Agrega o actualiza el valor de un atributo

element.setAttribute('src', "/images/cat.png")

1. hasAttribute: Retorna un booleano (true o false) si el atributo existe o no en el selector.

element.hasAttribute('src')

1. removeAttribute: Quita el atributo de un elemento

element.removeAttribute('src')

**Hablemos ahora de modificación de estilos**

Para realizar un cambio en el estilo de una etiqueta, utilizamos style, seguido de la propiedad que quieres ejecutar. Puedes ver el listado de las propiedades [aquí](https://www.w3schools.com/jsref/dom_obj_style.asp).

const p = document.querySelector('p')

p.style.fontSize = "26px"

**¡Hola! ¿Cómo estás? ¿Cómo te sientes? ¿Te parece si vemos un video para ejercitarnos mientras estamos en la PC? [presiona aquí para verlo.](https://www.youtube.com/watch?v=TNulqiGyEZU)**

**¿Hiciste el ejercicio? ¿Qué tal, mejor?**

**¡Sigamos adelante...!😎**

**Eventos**

Los eventos son acciones que pasan dentro de un navegador web y que le ocurren a elementos HTML, éstos son inherentes al navegador web en sí, el cual es el encargado de controlarlos y detonarlos cada vez que ocurra una acción por parte del usuario, por lo que cuando se implementan con JavaScript, se tienen acceso a ese conjunto de eventos y se pueden "capturar", lo que significa que el código JavaScript "reacciona" a los mismos, por lo que podemos manejar los eventos si se asigna un event handler a los elementos HTML a los cuales queremos capturarles los eventos cuando éstos ocurran.

Un evento puede ser ocasionado por una acción interna del código, del navegador web en sí o puede ser ocasionado por una acción del usuario. Por ejemplo, cuando termina de cargar una página, el navegador web automáticamente detona el evento onload, o cuando un usuario cambia información en una caja de texto, eso detona el evento onchange (y algunos otros más muy útiles como el onkeyup, onkeydown y onkeypress), o cuando se hace clic sobre un botón, se detona el evento onclick; todos éstos eventos se pueden capturar con un *listener* en JavaScript, ya que muy seguramente cuando un evento ocurra, se querrá hacer algo al respecto, por lo que el *listener* permite ejecutar un pedazo de código cuando un evento ocurre.

Los event handlers normalmente son asignados usando atributos de evento, esto es, atributos especiales de un elemento HTML con el que el navegador avisa que algún evento ocurrió sobre ese elemento. Normalmente los atributos de eventos empiezan con el prefijo on seguido de una referencia al evento que se quiere escuchar, los cuales son prácticamente autoexplicativos. Los más usados son:

* **onchange:** Ocurre cuando un elemento cambió de valor.
* **onclick:** Ocurre cuando el usuario dio clic sobre la superficie de un elemento.
* **onmouseover:** Ocurre cuando el usuario mueve el puntero sobre la superficie de un elemento.
* **onmouseout:** Ocurre cuando el puntero salió de la superficie de un elemento.
* **onkeydown:** Ocurre cuando el usuario presiona una tecla sobre un elemento que tenga el foco.
* **onload:** Ocurre cuando el navegador termina de descargar la página completa.

Hay muchísimos más eventos, pero éstos son unos de los más usados.

Por ejemplo, un event handler, puede ser un pedazo de código JavaScript que directamente se asigne en el atributo de evento (ejemplo: <button id="boton1" onclick="alert('Hola Mundo!')">Presióname</button>) o una función ya definida en JavaScript (ejemplo <button onclick="manejadorEvento()">Presióname</button>).

También se pueden asignar eventos usando el DOM directamente, ejemplo document.getElementById("boton1").onclick = manejadorEvento();.

Y también se pueden agregar varias funciones a un mismo evento, usando la propiedad addEventListener del DOM, al hacer esto se puede estar ejecutando 2 o más pedazos de código cada vez que un evento en particular ocurra, esto no sobrescribe asignaciones previas (como la del atributo de evento onclick), si no que cada vez que se asigne, se crea una ejecución:

<button \*onclick\*="alert('Hola Mundo!')">Presióname</button>

<h1 \*onclick\*="cambiarTexto(this)">Presiona el texto para cambiarlo</h1>

<script>

function cambiarTexto(id) {

id.innerHTML = "Texto cambiado";

}

function manejadorEvento() {

alert('Hola desde el event handler.');

}

function manejadorEvento2() {

alert('Hola desde el segundo event handler.');

}

const boton1 = document.getElementById('boton1');

boton.onclick = manejadorEvento();

element.addEventListener("click", manejadorEvento);

element.addEventListener("click", manejadorEvento2);

</script>

*Ejemplo de eventos en JavaScript*

**Te dejo por aquí una infografía sobre eventos, si quieres consultarla y descargarla [Presiona aquí.](https://github.com/U-Camp/BOOT-M1-SEM8/blob/main/infografias/M1_S8_Infografia%2001.pdf)**

**Hasta ahora, ¿cómo te sientes?, recuerda que nuestro cuerpo es muy importante, te dejo por aquí un video que te muestra algunos ejercicios que puedes hacer para mejorar tu espalda [presiona aquí.](https://www.youtube.com/watch?v=swtgR3OWfQg)**

**Nos vemos la próxima semana. 🕙**