«Talento Tech»

Desarrollo Web 2

Clase 07

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 7 | Introducción al DOM

Temario:

- Introducción al DOM
- Desde el DOM al HTML: getElementBy()
- QuerySelector

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Introducción al DOM

El **DOM** (Document Object Model) es una estructura jerárquica que representa un documento HTML o XML en una página web. Básicamente, el DOM es una representación del contenido de una página web en forma de árbol de nodos, donde cada elemento HTML se convierte en un objeto del DOM.

Es muy importante porque permite a las y los desarrolladores web interactuar y manipular los elementos HTML de una página web utilizando JavaScript. Es decir, a través del DOM, podemos acceder y modificar el contenido, atributos y estilos de los elementos de una página web.

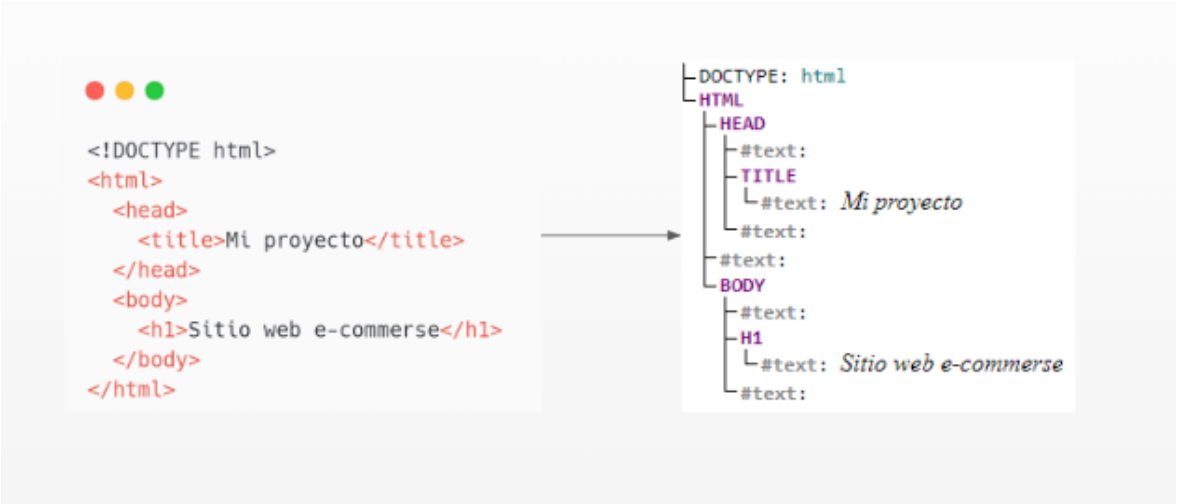
Por ejemplo, si queremos cambiar el texto de un párrafo en una página web, podemos utilizar JavaScript para acceder al objeto DOM que representa ese párrafo y modificar su contenido.

Para utilizar el DOM, simplemente se necesita escribir código JavaScript y hacer uso de las funciones y métodos que proporciona el lenguaje para interactuar con los objetos del DOM.

En otras palabras, el **DOM** es como un **mapa que el navegador crea para que podamos acceder y modificar los diferentes elementos presentes en la página HTML**. Esto nos permite realizar cambios en tiempo real, como por ejemplo, editar el texto que se muestra en una etiqueta <h1>. Así, con JavaScript podemos lograr una experiencia interactiva y dinámica para los usuarios que visitan nuestro sitio web.

En el **DOM**, cada etiqueta HTML es representada como un **objeto**, que también se conoce como **nodo**.
Cuando una etiqueta se encuentra dentro de otra etiqueta, decimos que está **anidada** y la llamamos "**nodo hijo**" de la etiqueta que la contiene, que a su vez es conocida como "**nodo padre**".
Esta **relación jerárquica** entre las etiquetas nos permite organizar y relacionar los diferentes elementos presentes en la página web, lo que facilita su manipulación y modificación dinámica mediante JavaScript

Todos estos objetos son accesibles empleando **JavaScript** mediante el objeto global **document**
Por ejemplo, **document.body** es el nodo que representa la etiqueta **<body>**.



Si queremos ver las propiedades y métodos detallados del documento, hacemos un console.dir(document);

Desde el DOM al HTML

Ahora que ya sabemos como acceder al **DOM**, el siguiente paso será elegir un elemento de este **árbol de elementos** que representa a todo el archivo **HTML** de la página web para que podamos manipularlo desde Javascript. Para seleccionar elementos del HTML desde Javascript tenemos varios métodos. Podemos elegirlos por su **id**, por su **class** o su **nombre** (li, div, section, h2, etc). Existen varios métodos para realizar esta selección.

getElementBy()

Es un método en JavaScript que se utiliza para seleccionar elementos HTML dentro de un documento. Este método toma como parámetro el nombre de la etiqueta HTML o el valor del atributo "id" que deseas seleccionar y devuelve un objeto que representa el elemento HTML seleccionado.

Por ejemplo:
Si tenés un elemento de tipo "p" con un "id" de "parrafo1", en tu documento HTML, puedes seleccionarlo usando **getElementById('parrafo1')**. Esto te devuelve un objeto que representa ese elemento HTML, el cual puedes manipular utilizando **JavaScript**.

Para visualizar los nodos creados desde **DOM** te recomendamos utilizar la extensión de Chrome 👉 DOM node tree viewer.

Una vez instalado debes seguir los siguientes pasos:
1) Abrir una nueva pestaña vacía con el inspector abierto
2) Dejando el inspector abierto, abrir una nueva página.
3) Ir a la parte superior derecha, al icono del rompecabezas, y seleccionar la extensión.
Automáticamente abrirá el árbol de nodos, el DOM, de todos los elementos que forman parte del sitio. Esto sólo funciona si el sitio está subido a un host.

Existen distintos métodos para acceder a los elementos del DOM empleando en la clase Document.

Los más comunes son:

- **getElementById()**: Sirve para acceder a un elementos específicos por su **ID**

```
let portadaInicio = document.getElementById("portada");
console.log(portadaInicio);
```

- **getElementsByClassName()** > Sirve para acceder a un conjunto de elementos de la estructura HTML, utilizando su atributo class como identificación. Se retornará un Array de elementos con todas las coincidencias.

```
let items = document.getElementsByClassName("liNav");
console.log(items);
```

- **getElementsByTagName()** : Sirve para acceder a un conjunto de elementos de la estructura HTML, utilizando su nombre de etiqueta como identificación. **Esta opción es la menos específica de todas**, ya que es muy probable que las etiquetas se repitan en el código HTML.

```
let caja = document.getElementsByTagName("div");
console.log(caja);
```

querySelector()

Se utiliza para seleccionar el primer elemento que coincide con el selector especificado en un documento HTML. Si hay varios elementos que coinciden, sólo se selecciona el primero. Para seleccionar un elemento con un ID específico, usarías **#nombre-id**, y para seleccionar un elemento por su clase, usarías **.nombre-clase.**

Supongamos que tenemos un elemento **<h2>** en **HTML** y quiero acceder a ella mediante javascript. Lo que se hace es:
Al ejecutar este método lo que hacemos es seleccionar el primero H2 que encuentre el HTML y ahora vamos a guardarlo en un constante para tenerlo siempre disponible en nuestro javascript.

```
const subtitulo = document.querySelector("h2");
```

Una vez que tenemos el elemento guardado en una constante, ya podemos acceder a sus propiedades. Algunas de ellas son:

- style
- classList
- setAttribute
- innerHTML
- textContent

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Style

Para modificar estilo de CSS o agregarlos desde javascript, podemos hacerlo accediendo a la propiedad style.

HTML:

```
<h2>Mi Subtitulo</h2>
```

JavaScript:

```
const h2 = document.querySelector("h2");
//Cambia el color
h2.style.color = "red";
//Cambia tipografía
h2.style.fontFamily = "Arial, Helvetica, sans-serif";
//Cambia tamaño de letra
h2.style.fontSize = "50px";
```

A tener en cuenta: Las propiedades que en CSS se dividían con un guión del medio (-) en **JavaScript** utilizan **camelCase**

- font-family = fontFamily
- font-size = fontSize
- background-color = backgroundColor

setAttribute, getAttribute

Vimos que muchas etiquetas HTML utilizan atributos: href, src, target, etc. Mediante Javascript se pueden acceder a estos atributos o modificarlos.

```
const link = document.querySelector("a");

link.getAttribute("href");

link.setAttribute(href, "https://www.google.com/");
```

textContent

Mediante javascript se puede acceder o modificar el contenido de una etiqueta, entendiendo por contenido a lo que está dentro de la etiqueta.

```
const h2 = document.querySelecctor("h2");
h2.textContent; //Devolverá el contenido del h2.
h2.textContent = "Nuevo contenido"; // Cambia el contenido
```

InnerText

Es similar a la de textContent, pero con este podemos agregar HTML a distintos nodos. Por ejemplo, si quisiéramos agregar un dentro de un <h2>, con este método eso sería posible.

```
const titulo = document.querySelector("h1");

titulo.innerHTML; //Devuelve el contenido del h1 (con sus etiquetas)



titulo.innerHTML = <span> Span dentro del h1 </span>;

/*Agregará nuevo contenido en HTML al h1,

    eliminando el contenido anterior.*/
```

Cuando trabajamos con **document.querySelector()**, no siempre tenemos que seleccionar elementos por su etiqueta, sino que podemos hacerlo mediante clases, id e incluso especificidad.

```
document.querySelector("h2"); //Selecciona por elemento

document.querySelector(".titulo-principal"); //Selecciona por clase

document.querySelector("#titulo-principal"); //Selecciona por id

document.querySelector("header nav .nav-li"); //Selecciona por
especificidad
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

❓ Pregunta de reflexión

¿En qué situaciones sería más apropiado utilizar **querySelector()** en lugar de **getElementById()** para seleccionar elementos del DOM?

¿Cómo podría el uso del DOM y JavaScript permitir la creación de una experiencia de usuario más dinámica en una página web? Piensa ejemplos específicos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

📝Desafío #7

Crea un archivo HTML básico con la siguiente estructura:

```
<body>

    <h1 id="nombreProducto">Smartphone XYZ</h1>
    <p id="precioProducto">Precio: $500</p>


    <script src="app.js"></script>
</body>
```

En este desafío, simularemos un cambio de precio en un producto de un e-commerce.

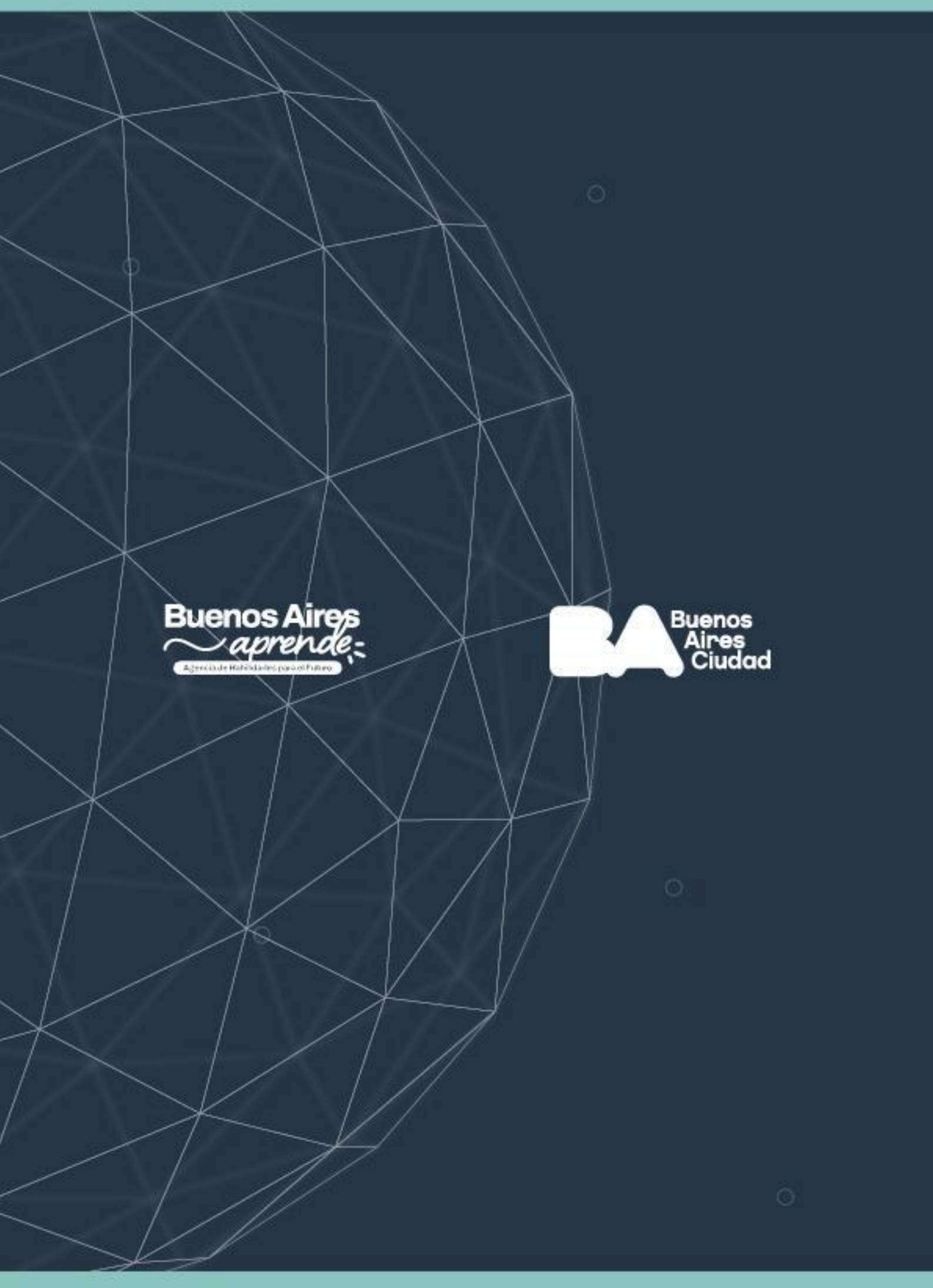
Al cargar la página, queremos que el precio inicial de "$500" cambie a "$450" utilizando las propiedades **getElementById()** e **innerText.**

También podemos modificar los estilos del título mediante querySelector como en el siguiente ejemplo donde le cambiamos el color al título.

```
// Selecciona el elemento del precio por su id
let precioProducto = document.getElementById("precioProducto");


// Cambia el texto del precio
precioProducto.innerText = "Precio: $450";

const titulo = document.querySelector("h1");
titulo.style.color = "red";
```


Buenos Aires
aprende
Buenos
Aires
Ciudad