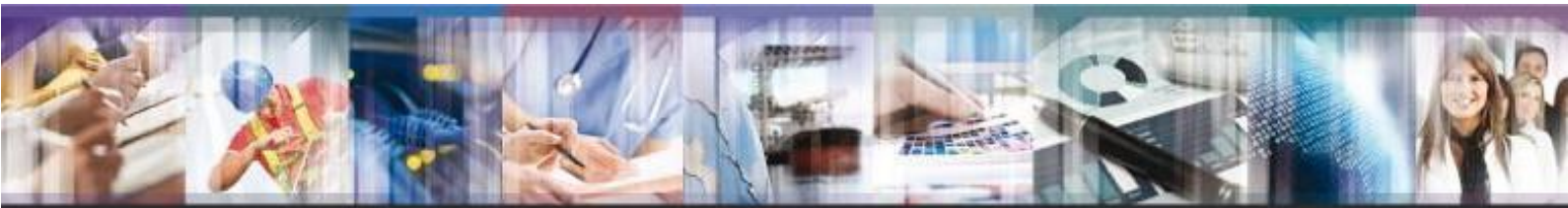
UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

Temario:

Temario:



1. ¿Qué es el DOM y para qué lo utilizamos?

El DOM (Document Object Model, o en español Modelo de Objetos de Documentos) es una interfaz entre JavaScript y HTML o XML (otro metalenguaje que no abordamos en este curso). Nos permite acceder a sus partes y propiedades e ir interactuando con ellas. Tenemos que tener en cuenta que cuando hablamos de propiedades de las partes del HTML estamos hablando de su CSS por lo que el DOM nos va a permitir modificar el CSS también, por medio de HTML.

El DOM es una de las interfaces más utilizadas en la Web y surgió a partir de la utilización de JavaScript, permitiéndonos agregar comandos en el navegador para manipular el HTML según vayamos necesitando.

Los objetos del DOM modelizan tanto la ventana del navegador como el historial, el documento o página web y todos los elementos que pueda tener dentro la propia página, como párrafos, divisiones, tablas, formularios y sus campos, entre otros. A estos elementos a manipular los llamaremos ***nodos.***

1.1 Nodos

Teniendo en cuenta lo que indicamos anteriormente, podemos definir como Nodo a cualquier etiqueta del cuerpo del HTML.

Si vemos cómo se compone el HTML, podemos ver que tiene elementos padres que contienen a elementos hijos, y estos pueden contener a su vez a otros elementos.Esta sucesión de elementos anidados se denomina árbol de nodos.

Teniendo en cuenta esto, al acceder a un nodo padre podemos acceder a las propiedades que afectan a los nodos hijos.

1.2 Accediendo al DOM

Para acceder al DOM necesitamos desde nuestro archivo JavaScript utilizar las palabras reservadas ***document*** o ***window***, las que reconoce la interfaz para hacer referencia a la hoja HTML a la que está asociado ese archivo JavaScript. Junto con esta palabra reservada necesitamos utilizar algunas funciones que nos provee JavaScript para poder indicar el nodo al que queremos acceder. Ellos son:

- **getElementById():** Esta función nos permite acceder a un elemento indicando su id. Al ser un nombre único en todo el documento HTML, nos devolverá un único elemento.

SINTAXIS

document.getElementById("nombreDeId")

EJEMPLO

var parrafoParticular = document.getElementById("texto")

- **getElementByClassName():** Esta función nos permite acceder a elementos que contengan la clase que le indiquemos, devolviéndonos una lista de nodos. Para acceder a cada uno de sus valores, vamos a utilizar las propiedades vistas de array.

SINTAXIS

document.getElementByClassName("nombreDeLaClase")

EJEMPLO

var tarjetas = document.getElementByClassName("cards")

- **getElementByTagName():** Esta función nos permite acceder a un elemento que comparta el tipo de etiqueta. En este caso también vamos a tener como resultado una lista de nodos, ya que podemos tener varios elementos del mismo tipo, por lo que utilizaremos las propiedades de array para acceder a cada valor recibido.

SINTAXIS

document.getElementByTagName("nombreDeLaEtiqueta")

EJEMPLO

var titulos = document.getElementByTagName("h2")

- **querySelector():** Esta función nos permite acceder al primer elemento que cumpla con el selector que le pasemos. Recordemos que los selectores nos permiten hacer referencia a distintos tipos de elementos, ya sea el símbolo numeral (#) para indicar que se trata de un id, el punto (.) para indicar que se trata de una clase, y el nombre solo cuando indicamos un elemento.

En este caso, tenemos que tener en cuenta que solamente nos va a devolver un **único nodo**, el primero que se encuentre (leyendo de izquierda a derecha, de arriba hacia abajo) en el HTML.

SINTAXIS

document.querySelector("selector")

EJEMPLO

var primeraTarjeta = document.querySelector(".cards")

- **querySelectorAll():** Esta función es una variante de la función querySelector que nos permite acceder a todos los nodos que cumplen con el selector que le pasamos. Esto nos va a devolver una lista de nodos por lo que haremos uso de las propiedades de array vistas para poder acceder a cada uno de los nodos recibidos.

SINTAXIS

document.querySelectorAll("selector")

EJEMPLO

var todasLasTarjetas = document.querySelectorAll(".cards")

Teniendo en cuenta esto, podemos notar que en los ejemplos guardamos a los nodos invocados en variables. Esto nos va a permitir poder utilizarlos a lo largo de todo el archivo JavaScript de una forma simple y cómoda, haciendo uso de una buena práctica.

2.Modificando el DOM

Una vez que ya pudimos acceder al nodo o a los nodos que necesitamos, podemos manipularlos según lo que necesitemos hacer.
Para ello utilizaremos distintos métodos que definiremos a continuación según la acción que vayamos a realizar.

2.1 Creación de nuevos elementos

Para poder crear un nodo, contamos con el método ***createElement()***, que nos pide que le indiquemos la etiqueta a crear.

SINTAXIS

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

Una vez creado ese elemento, podemos añadirlo a otro nodo padre utilizando el método ***appendChild()***, donde le indicamos a qué padre va a estar asociado dicho nodo que creamos.

SINTAXIS

```
nodoPadre.appendChild(nodoHijo)
```

EJEMPLO

```
var titulo = document.getElementById("contenedor-titulo-principal");

var  textoNuevo= document.createElement("p");

titulo.appendChild(textoNuevo)
```

En este caso, tenemos que tener en cuenta que va a agregar al elemento antes del cierre del elemento padre. En caso que queramos agregar a dicho elemento en otro lugar del padre,

podemos utilizar el método ***insertBefore() ,*** que requiere al nodo padre para saber en qué nodo se llevará a cabo este inserción, y dos argumentos: el nodo a insertar y el nodo hijo de referencia para saber delante de qué nodo hijo se ubicará este nodo creado.

SINTAXIS

nodoPadre.insertBefore(nodoHijoNuevo,nodoHijoDeReferencia)

EJEMPLO

```
var titulo = document.getElementById("titulo-principal");

var texto1 = document.querySelector("#parrafo1");

var  textoNuevo= document.createElement(p);

titulo.insertBefore(textoNuevo, texto1 )
```

2.2. Modificando nodos existentes

Otra de las cosas que podemos hacer es sacar nodos que se encuentren en el HTML. Para ello, podemos hacer uso del método ***removeChild()*** el cual nos pide que indiquemos tanto el nodo padre para posicionarse en un lugar específico como el nodo hijo, para saber el nodo a remover.

SINTAXIS

nodoPadre.removeChild(nodoHijo)

EJEMPLO

```
var titulo = document.getElementById("titulo-principal");

var texto1 = document.querySelector("#parrafo1");

titulo.removeChild(texto1)
```

También podemos reemplazar un nodo por otro utilizando el método ***replaceChild()*** que necesita saber cuál es el nodo padre donde este cambio va a surgir, y dos argumentos: el nodo a insertar y el nodo a reemplazar.

SINTAXIS

nodoPadre.replaceChild(nodoNuevo,nodoAReemplazar)

EJEMPLO

```
var titulo = document.getElementById("titulo-principal");

var texto1 = document.querySelector("#parrafo1");

var  textoNuevo= document.createElement(p);

titulo.replaceChild(textoNuevo,texto1);
```

También podemos utilizar las siguientes propiedades para poder acceder a contenido de nodos y modificarlos. Ellos son:

- **textContent:** Esta propiedad nos permite acceder al contenido de un elemento e incluso poder reemplazarlo con uno nuevo.

SINTAXIS

Para acceder al valor

var nombreVariable = nodo.textContent

Para reemplazarlo con otro

nodo.textContent = nuevoContenidoEnFormatoString

EJEMPLO

- **innerHTML:** Esta propiedad nos permite acceder al contenido de nodos HTML que contiene un nodo padre e incluso insertar un nuevo nodo al HTML.

SINTAXIS

Para acceder al valor

var nombreVariable = nodo.innerHTML

Para reemplazarlo con otro

nodo.innerHTML = "nuevoNodoAIngresar"

EJEMPLO

2.3 Modificación de propiedades de nodos

Además de poder manipular los nodos, contamos con la posibilidad de modificar propiedades que tengan los nodos. Para ello, contamos con una amplia variedad de métodos que nos permiten acceder a propiedades existentes y crear propiedades para los métodos según lo necesitemos.

En este módulo, veremos los más utilizados.

- **getAttribute:** Este método nos permite obtener el atributo del nodo indicado.

SINTAXIS

nodo.getAttribute("nombreDelAtributo")

- **setAttribute:** Este método nos permite asignar un atributo a un nodo indicado.

SINTAXIS

nodo.setAttribute("nombreDelAtributo" , "nuevoValorDelAtributo")

EJEMPLO

- **removeAttribute:** Este método nos permite eliminar o remover un atributo de un nodo

indicado.

SINTAXIS

nodo.removeAttribute(“nombreDelAtributo”)

- **classList.add:** Este método nos permite agregar una clase al nodo. De esta forma podemos definir todas las propiedades para un estilo y agregarlo en el momento que necesitemos al nodo correspondiente.

SINTAXIS

nodo.classList.add(“nombreDeLaClase”)

EJEMPLO

- **classList.remove:** Este método nos permite eliminar una clase que contenga un nodo, lo que nos va a servir en caso de que en algún momento del comportamiento del sitio necesitemos que no cuente con alguna clase de nodo.

SINTAXIS

nodo.classList.remove(“nombreDeLaClase”)

- **classList.toggle:** Este método nos permite agregar o eliminar una clase de un nodo, dependiendo de si dicho nodo contiene o no la clase. Esto quiere decir que se va a fijar si ese nodo tiene esa clase y, en caso de que la tenga, la va a remover; caso contrario, la va a agregar.

SINTAXIS

nodo.classList.toggle(“nombreDeLaClase”)

EJEMPLO

- **style:** Este atributo nos permite acceder, agregar o reemplazar cualquiera de las propiedades de CSS en el nodo que indiquemos. Debemos indicar la propiedad de CSS con la escritura camelCase en caso que la propiedad en CSS se escriba con guión. Es decir, en el caso de utilizar a la propiedad ***font-weight*** escribiremos ***fontWeight*** ; si queremos utilizar la propiedad ***text-decoration*** escribiremos ***textDecoration***, etc.

SINTAXIS

Para acceder al valor

var nombreVariable = = nodo.style.propiedadCSS

Para reemplazarlo con otro

nodo.style.propiedadCSS = "nuevoValorDeLaPropiedad"

EJEMPLO

3.En resumen

En esta unidad pudimos aprender de una nueva herramienta: el DOM, que nos permite interactuar entre los distintos lenguajes que maneja el navegador (HTML-CSS-JavaScript).

También aprendimos lo que son los nodos y a qué denominamos árbol de nodos.

Por otra parte, pudimos ver los distintos métodos que nos ofrecen, utilizando el lenguaje JavaScript para acceder, reemplazar y eliminar tanto nodos como propiedades y contenido.

4.Bibliografía utilizada y recomendada

https://developer.mozilla.org/es/docs/Glossary/DOM
https://desarrolloweb.com/articulos/que-es-el-dom.html
https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction

https://developer.mozilla.org/es/docs/Web/API/Document/querySelector
https://developer.mozilla.org/es/docs/Web/API/Document/createElement