COLOMBIA POTENCIA DE LA VIDA

TIC

Módulo 1

LECCIÓN 3

Manipulación del DOM, atributos de los elementos HTML

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

Universidad Tecnológica de Pereira

Selectores

Si se encuentra con un código Javascript y se quiere hacer modificaciones en un elemento de la página HTML, lo primero que se debe hacer es buscar dicho elemento. Para ello, se suele intentar identificar el elemento a través de alguno de sus atributos más utilizados, generalmente el id o class.
Existen una serie de métodos que nos permitirán buscar en el DOM de la página y encontrar dichos elementos. El primer grupo son métodos más antiguos y tradicionales, el segundo grupo son métodos más modernos, pero requieren conocer selectores CSS.
Métodos tradicionales

Los métodos más clásicos y tradicionales para realizar búsquedas de elementos en el DOM son más sencillos, pero menos potentes. Si lo que buscas es un elemento específico, lo mejor sería utilizar el método getElementById(). En caso contrario, utilizaremos alguno de los otros tres métodos, que nos devuelven siempre un array:

.getElementById(id): Busca el elemento HTML por su id. **.getElementById(id)** busca un elemento HTML con el id especificado. En principio, un documento HTML bien construído no debería tener más de un elemento con el mismo id, por lo tanto, este método devolverá siempre un solo elemento

JavaScript

```
const element = document.getElementById("intro");
// <div id="intro"></div>
```

.getElementsByClassName(class): Busca elementos con la clase class Es importante darse cuenta del matiz de que el método tiene getElements en plural, es decir, puede devolver varias clases ya que al contrario que los id, pueden existir varios elementos con la misma clase

JavaScript

```
const x = document.getElementsByClassName("intro");
// <div class="intro"></div>
// <div class="intro"></div>
// <div class="intro"></div>
```

.getElementsByName(value): Busca elementos con el atributo name a value

.getElementsByTagName(tag): Busca etiquetas HTML tag.

Métodos modernos

Aunque los métodos tradicionales anteriores son más que suficientes para buscar elementos, actualmente tenemos a nuestra disposición dos nuevos métodos de búsqueda de elementos que son mucho más cómodos y prácticos si conocemos y dominamos los selectores CSS.

El primero devuelve un único elemento mientras que el segundo devuelve un array de elementos:

- **.querySelector(sel):** Busca el primer elemento que coincide con el selector CSS sel.
- **.querySelectorAll(sel):** Busca todos los elementos que coinciden con el selector CSS sel.

Recuerden los selectores base de CSS, de clase (.), de id (#) y de etiqueta.

Ahora que se han capturado elementos del DOM con los métodos vistos, ¿qué atributos tengo disponibles en la etiqueta?

¿Cómo creen ustedes que puedo ver las opciones que me entrega un elemento capturado del DOM como un título o un div por ejemplo?

Element - Web APIs | MDN

localStorage and sessionStorage

La función de ambos es almacenar datos en el navegador de cada usuario, y podemos utilizarla desde cualquier vista.
Las dos funcionan exactamente igual, con los mismos métodos. La diferencia está en cuánto tiempo se guarda la información.

localStorage
Permite almacenarlas por tiempo indeterminado.
sessionStorage
Las guarda en sesión: si cerramos el navegador, se pierde la info.

Métodos de storage

setItem

Unset

```
sessionStorage.setItem('nombre', 'Dario');
```

Este método crea nuevos atributos al storage y asigna atributos para ellas, recibe dos parámetros: la clave y el valor.

getItem

Unset

```
sessionStorage.getItem('nombre'); //Dario
```

Nos devuelve el valor de la clave que le pasemos.

removeItem

```
Unset

sessionStorage.removeItem('nombre');
```

Recibe como parámetro la clave, y la busca dentro de storage para eliminarla.

clear

```
Unset

sessionStorage.clear();
```

Borra todo el contenido que se haya almacenado en storage.

Ejemplo

```
Unset

window.addEventListener('load', ()=>{

        //le pedimos por prompt al usuario que guarde su nombre

        let nombre = prompt('Dinos tu nombre');

        //seleccionamos en el HTML un <p> para agregarle esa info
```

```
dobument.querySelector('.bienvenida').innerHT
ML = "Hola " + nombre;

        //Hasta acá, si recargamos la página o
cerramos el navegador, va a volver a pedir

        //el nombre

        localStorage.setItem('nombreUsuario',
nombre)

        //esto se guarda y queda por más que borre
la línea

};
```

Para no sobrescribir o generar un error, usamos un condicional

Unset

```
window.addEventListener('load', ()=>{



//le preguntamos si había algo en localStorage, y
en caso de que no (null), le pida

//el nombre y lo guarde
```

```
if(localStorage.getItem('nombreUsuario') == null){
                let nombre = prompt('Dinos tu nombre');

document.querySelector('.bienvenida').innerHTML = "Hola " + nombre;

localStorage.setItem('nombreUsuario', nombre)

        //así como está no va a imprimir el nombre en la pantalla, eso lo hacemos en el else

        } else {

                let nombre = localSotrage.getItem('nombreUsuario');

document.querySelector('.bienvenida').innerHTML = "Hola " + nombre;


        };

};
```

En storage sólo podemos guardar variables de TEXTO por lo cual si estamos trabajando con array o con objetos literales, y queremos guardar toda esa información en storage, habría que pasar la información por **JSON.stringify**, lo que nos va a permitir transformar ese array u objeto en un texto, siendo una estructura que sí se puede guardar en el storage.

Ejercitación

Local storage

Tenemos que capturar el item de local storage con el nombre "id", que sabemos se creó en otro script desarrollado por un compañero, e imprimirlo en el <h1> con id "idValue".

Para esto tenemos que:

- Guardar el valor del item "id" en la variable "id".
- Capturar, en la variable idValue, el <h1> con id idValue.
- Asignarle al texto del <h1> el valor de la variable id.

Unset

```
window.addEventListener('load', function(){

  //TU CODIGO AQUI

  let id = localStorage.getItem("id")

  let idValue =
document.querySelector("#idValue")

  idValue.innerHTML = id



})
```

Session storage

Nos piden que creemos en sessionStorage un item con el nombre "bgColor" con el valor "red" para que desde otro script puedan cambiar el color del body. Además, nos piden que eliminemos el item "font".

```
Unset
window.addEventListener('load', function(){



//TU CODIGO AQUI

//

  localStorage.setItem("bgColor", "red");

  localStorage.removeItem("font")

})
```

Ejercitación

Realizar un gestor de tareas, debe crear un aplicativo web que le permita al usuario registrar sus tareas pendientes, las tareas deben contar con un título y una descripción, para ello debe implementar un formulario con al menos esos dos campos.
Una tarea que se crea, debe ser guardada en el almacenamiento local y debe aparecer en un listado de tareas pendientes debajo o a un costado del formulario de creación de tareas.
Una tarea creada puede ser borrada mediante un botón, cuando una tarea es borrada, ya no debe aparecer en el listado de tareas pendientes.

Reto extra:
Las tareas creadas pueden ser marcadas como completadas utilizando un botón, las tareas completadas deben mostrarse en un listado de tareas completadas debajo o a un costado del listado de tareas pendientes.