DOM y querySelector

Document Object Model

querySelector()

aprendé programando Virtual
SECUNDARIA del futuro
UTN F r a
GyE Gestión y Evaluación Educativa

DOM - ¿Qué es?

Ya venimos trabajando con JavaScript. Ya incorporamos lógica y pudimos resolver varios problemas. Sin embargo aún queda poder manipular con JavaScript, elementos del HTML y para poder hacer eso **vamos a hacer uso del DOM.**

Es un *conjunto de métodos, propiedades y eventos capaces de lograr interacción con el HTML.* La idea detrás de esto es poder, mediante JavaScript, interactuar con el HTML. Podemos definirlo como la forma en que JavaScript puede acceder al HTML.

Cambio de estructura, contenido, estilo

Haciendo uso del DOM vamos a poder cambiar el HTML y empezar a trabajar con interacción del usuario.

- Cuando el usuario hace click, que pase algo y se modifique la página.

DOM - ¿Qué es?

Ya sabemos lo que son objetos. El DOM en última instancia va a darnos un arbol de nodos, que dentro van a tener objetos con propiedades...

Para ver de qué estamos hablando, hagamos en la consola un **console.dir(document)** y recorramos algunos nodos.

Vamos a encontrarnos con que vemos algo similar a un objeto, que podemos recorrer y ver qué propiedades tiene.

En este caso, demos que el document tiene un montón de propiedades y una de ellas se llama **children.**

Vamos a ver que **children** tiene una ***collection*** de HTML. Pensemos en una collection como si fuera un array. Es decir que la propiedad children tiene, en este caso, un elemento que se llama [html]. Si abrimos este elemento (llamado **nodo**) vamos a ver que tiene muchas propiedades

DOM - ¿Qué es?

Una de esas propiedades es, a su vez, children.

```
▶ children: HTMLCollection(2) [head, body]
```

Esta vez vemos que el **nodo html** tiene dos children, **head y body.**

Y si ahora abrimos el nodo, por ejemplo, de head... ¿Con qué nos encontramos?

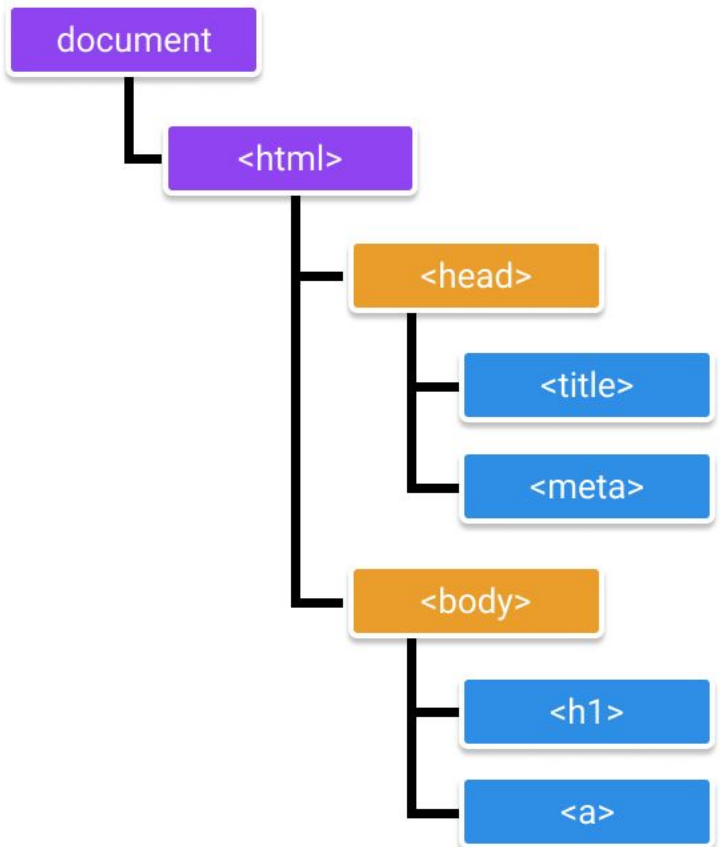
```
▼ children: HTMLCollection(4)
  ▶ 0: meta
  ▶ 1: meta
  ▶ 2: meta
  ▶ 3: title
    length: 4
  ▶ viewport: meta
  ▶ __proto__: HTMLCollection
```

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
```

Vamos a ver que tiene como nodos a los mismos elementos que hay dentro del head en el HTML.

DOM - ¿Qué es?

Como conclusión podemos decir que el DOM es una estructura de árbol que contiene todos los elementos que creamos en el HTML



Nuevamente vemos que en esta estructura de árbol, podemos acceder a los elementos html.

La pregunta ahora es... ¿Cómo los modificamos desde JavaScript?

document.querySelector

El objeto document tiene un montón de propiedades y métodos a utilizar. Uno de ellos se llama querySelector y funciona para seleccionar elementos HTML.

Supongamos que tenemos una etiqueta **<h2>** en nuestro HTML. Si quiero acceder a ella mediante JavaScript, lo que voy a hacer es:

document.querySelector

Ejecutando este método, lo que hacemos es seleccionar **el primer h2 que encuentre en el HTML** y ahora vamos a guardarlo en una constante para tenerlo siempre disponible en nuestro JavaScript.

```
const h2 = document.querySelector('h2');
```

Una vez que tenemos el h2 guardado, podemos acceder a propiedades del h2. Algunas de ellas son:

- classList
- style
- setAttribute
- setAttribute
- innerHTML
- textContent

style

Para agregar estilos CSS o modificarlos mediante JavaScript, podemos acceder mediante la propiedad style.

const h2 = document.querySelector('h2');

```
h2.style.color = "red";
h2.style.fontFamily = "Arial, Helvetica, sans-serif";
h2.style.fontSize = "50px";
```

El procedimiento es:

1. Selecciono la etiqueta a modificar
2. Accedo al estilo mediante su propiedad style
3. Cambio el estilo **igualándolo** al valor que necesitamos.

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

classList

Para agregar, remover o togglear clases, se utiliza la propiedad classList disponible en el nodo HTML.

const h2 = document.querySelector('h2');

h2.classList.add('claseAAgregrar');

h2.classList.remove('claseAEliminar');

h2.classList.toggle('miClase');

El procedimiento es:

1. Selecciono la etiqueta a modificar
2. Agrego/remuevo/toggleo mediante su propiedad classList y el método correspondiente.

Existen tres métodos dentro de la propiedad classList:

add()
remove()
toggle()

Por lo general, los estilos no los vamos a manipular mediante JavaScript sino que lo que vamos a manipular son las clases que agregamos y sacamos, y eso va a tener el CSS que querramos togglear.

setAttribute, getAttribute

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

const link = document.querySelector('a');

link.getAttribute("href");

Este método me devuelve el texto dentro del href de la primer etiqueta <a> que encuentre.

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

Este método recibe dos parámetros. El primero el atributo a modificar, el segundo el valor a modificar.

Es posible que al usar el método de getAttribute, querramos guardarlo en algún lado para luego utilizarlo. En ese caso simplemente lo guardamos en una variable.

const href = link.getAttribute("href");

textContent

Mediante JavaScript se puede acceder o modifical el contenido de una etiqueta, entendiendo por contenido lo que está dentro de la etiqueta.

const h2 = document.querySelector('h2');

h2.textContent;

Devolverá el contenido del h2

h2.textContent = "Nuevo contenido";

Agregará nuevo contenido al h2, eliminando el contenido anterior.

Es posible que al usar el método de textContent, querramos guardarlo en algún lado para luego utilizarlo. En ese caso simplemente lo guardamos en una variable.

const content = h2.textContent;

innerHTML

Esta propiedad es similar a la de textContent, con la diferencia de que podemos agregar HTML a distintos nodos. Por ejemplo si quisiéramos agregar un dentro de un <h2>, podríamos

const h2 = document.querySelector('h2');

h2.innerHTML;

Devolverá el contenido en html del h2 (con sus etiquetas)

h2.innerHTML = "Span dentro del h2";

Agregará nuevo contenido html al h2, eliminando el contenido anterior.

Es posible que al usar el método de innerHTML, querramos guardarlo en algún lado para luego utilizarlo. En ese caso simplemente lo guardamos en una variable.

const h2HTML = h2.innerHTML;

document.querySelector

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

```
document.querySelector('h2');

document.querySelector('.titulo-principal');

document.querySelector('#titulo-principal');

document.querySelector('header nav .nav-li');
```