Fundamentos JavaScript
Sesión 2 Semana 1

¿ Qué es el DOM ?

Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM (*o simplemente DOM*). Así que cuando nos referimos al DOM, estamos hablando de dicha estructura que podemos modificar de forma dinámica añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc.

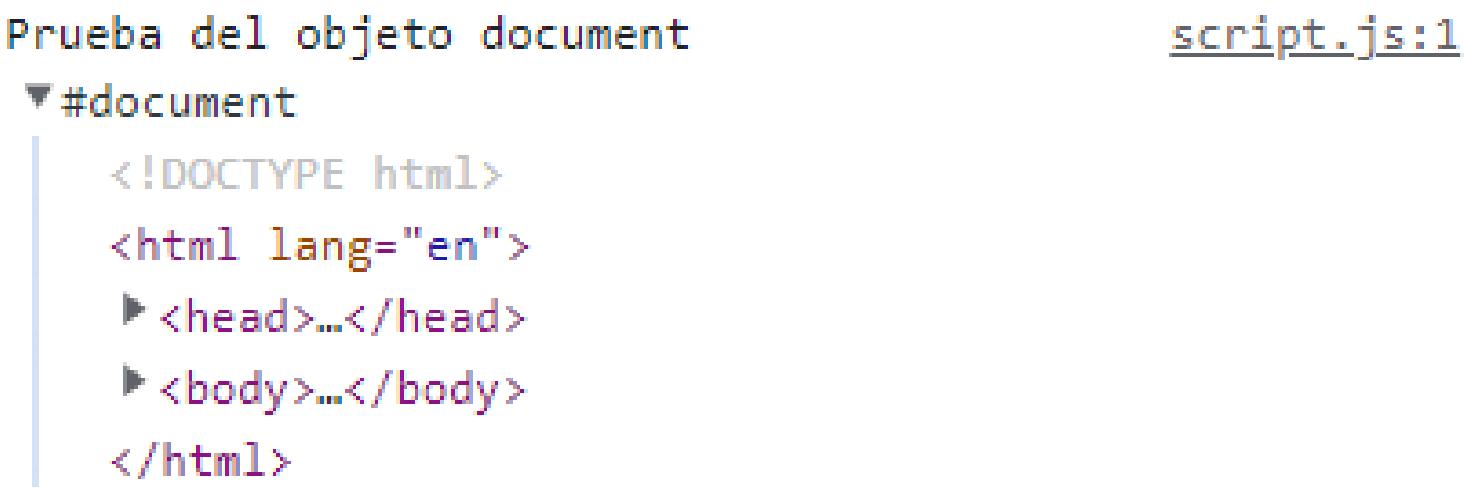
¿Cómo accedemos al DOM?

El objeto **DOCUMENT** es el que nos permite acceder a TODOS LOS ELEMENTOS DEL DOM,

ejemplo:

```
console.log("Prueba del objeto document", document);
```

Resultado



Nos muestra todo el contenido del archivo HTML

que tenemos enlazado

Seleccionar elementos del DOM

1. **getElementById("id del elemento"):** Cuando necesitamos utilizar un elemento en específico, utilizamos este método, que realiza una búsqueda por todo el DOM hasta conseguir el ID especificado (DEBIDO A QUE EL ID ES UN IDENTIFICADOR ÚNICO, SOLO RETORNARÁ UN ELEMENTO)

NOTA: El ID del elemento a buscar SIEMPRE se coloca dentro de los paréntesis, y este se recibirá como un STRING.

HTML:

```html
<div id="container"></div>
```

JS:

```
let contenedor = document.getElementById("container");

console.log(contenedor)
```

CONSOLA:

Podemos observar que mediante este método, logramos capturar el elemento del DOM que tenía el id “container”

2. **querySelector(“id, clase o etiqueta”):** Este método nos permite capturar un elemento específico, sin embargo, podemos buscar mediante el ID, clase o etiqueta del elemento, solo retorna el primer elemento que coincida con el parámetro de búsqueda especificado dentro de los paréntesis(dicho parámetro debe ser un string)

NOTA: Si vamos a buscar un elemento por su ID, debemos utilizar su selector **(“#idDelElemento”)**, de igual forma si vamos a buscar por su clase **(“.claseDelElemento”)**; y si solo vamos a buscar una etiqueta, se coloca el nombre de la misma (“div”)

HTML:

```
 <!-- etiqueta con id  -->
<div id="container"></div>

<!-- etiqueta con clase  -->
<h3 class="subtitulo"></h3>

<h1></h1>
```

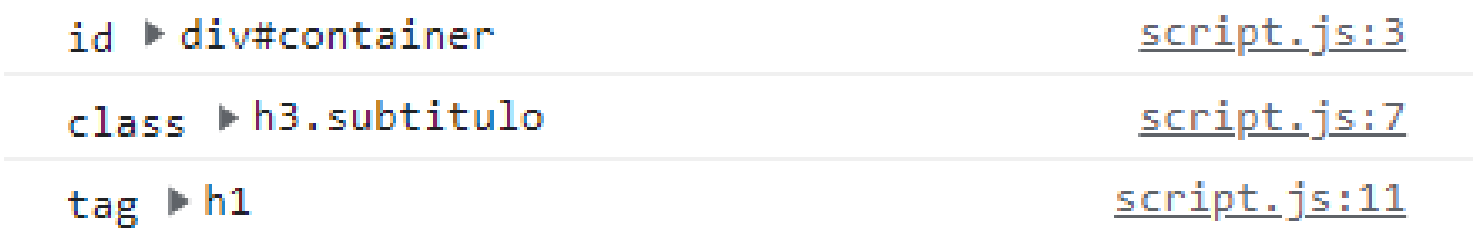
JS:

```
// Seleccionando un elemento por su ID
let getId = document.querySelector("#container");
console.log("id", getId);

//Seleccionando un elemento por su clase
let getClass = document.querySelector(".subtitulo");
console.log("class", getClass);

//Seleccionando un elemento por su etiqueta
let getTag = document.querySelector('h1');
console.log("tag", getTag)
```

CONSOLE:



RECORDEMOS...

El querySelector solo retorna el primer elemento que coincida con el parámetro de búsqueda que le especificamos

Otros métodos para seleccionar elementos del DOM

- .getElementsByClassName(class)
- .getElementsByName(name)
- .getElementsByTagName(tag)
- .querySelectorAll(sel)

Métodos para manipular el DOM

- **setAtributte(“atributo”, “valor”):** Nos permite actualizar los atributos de una etiqueta, y si ese atributo no existe, lo crea con el valor que le coloquemos

HTML:

```
<!-- Imagen sin el atributo src -->
<img alt="perrito">
```

JS:

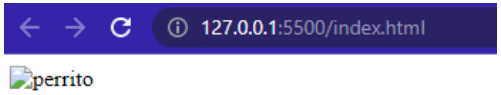
```
// Seleccionamos el elemento
let image = document.querySelector("img");

// Actualizamos el atributo src
image.setAttribute('src', 'https://cdn2.excelsior.com.mx/
```

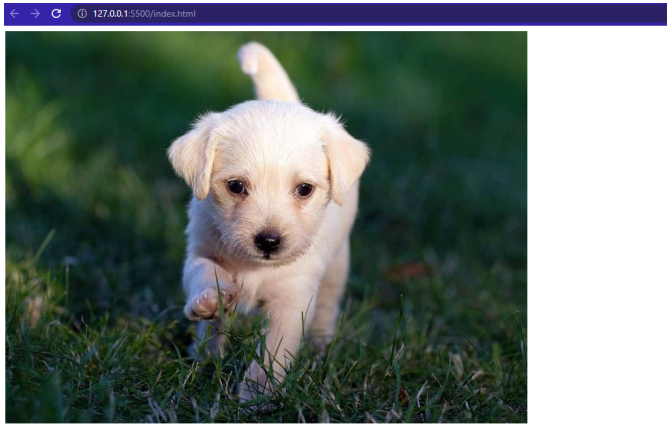
Como se observa se coloca la variable donde se encuentra el elemento, seguidamente del método. En el primer parámetro especificamos el nombre del atributo y en el segundo colocamos el valor que este tendrá

Resultado:

Antes



Después



Con este método podemos actualizar **cualquier atributo** que deseemos

Manipulando el CSS desde JS

HTML:

```
<body>

    <h1>Hola</h1>

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

JS:

```
// Capturamos el elemento
let body = document.querySelector("body");


//Actualizando diferentes estilos
body.style.background = "pink";
body.style.fontFamily = "Helvetica";
body.style.textAlign = "center"
```

Resultado:

Métodos importantes para la manipulación del CSS desde JS

- **classList:** nos permite listar todas las clases que contiene un elemento

- **classList.add("nombre de la clase a agregar"):** Nos permite agregarle una clase a un elemento.

- **classList.remove("nombre de la clase a eliminar"):** Nos permite eliminar una clase de un elemento.

- **classList.contains(“nombre de la clase”)**: retorna true si la clase existe en el elemento, de lo contrario retorna false

- **classList.replace (“primer clase”, “segunda clase”)**: Nos permite reemplazar clases en un elemento, la primera posición nos indica el nombre de la clase que vamos a reemplazar, y la segunda nos indica el nombre de la clase por el cual será reemplazada

Crear elementos en el DOM

- **createElement("etiqueta a crear"):** Nos permite crear una etiqueta, sin embargo, esta se crea **TOTALMENTE VACÍA.**

```
let image = document.createElement('img');
```

Insertar elementos en el DOM

- **appendChild(elemento):** Permite añadir un elemento al final de un contenedor (solo recibe valores de tipo node)

HTML:

```
<body>

    <div id="container"></div>


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

</html>
```

JS:

```
1   // creamos una imagen
2   let image = document.createElement('img');
3   // capturamos ek contenedor
4   let contenedor = document.getElementById('container');
5
6   //actualizamos el atributo src
7   image.setAttribute('src', 'https://www.tooltyp.com/wp-cont
8
9   // al contenedor seleccionado insertamos la
10  // etiqueta creada en la linea 2
11  contenedor.appendChild(image);
12
```

- **innerHTML:** Permite agregar o reemplazar el contenido de un elemento padre, este método recibe código **HTML EN FORMA DE STRING**.

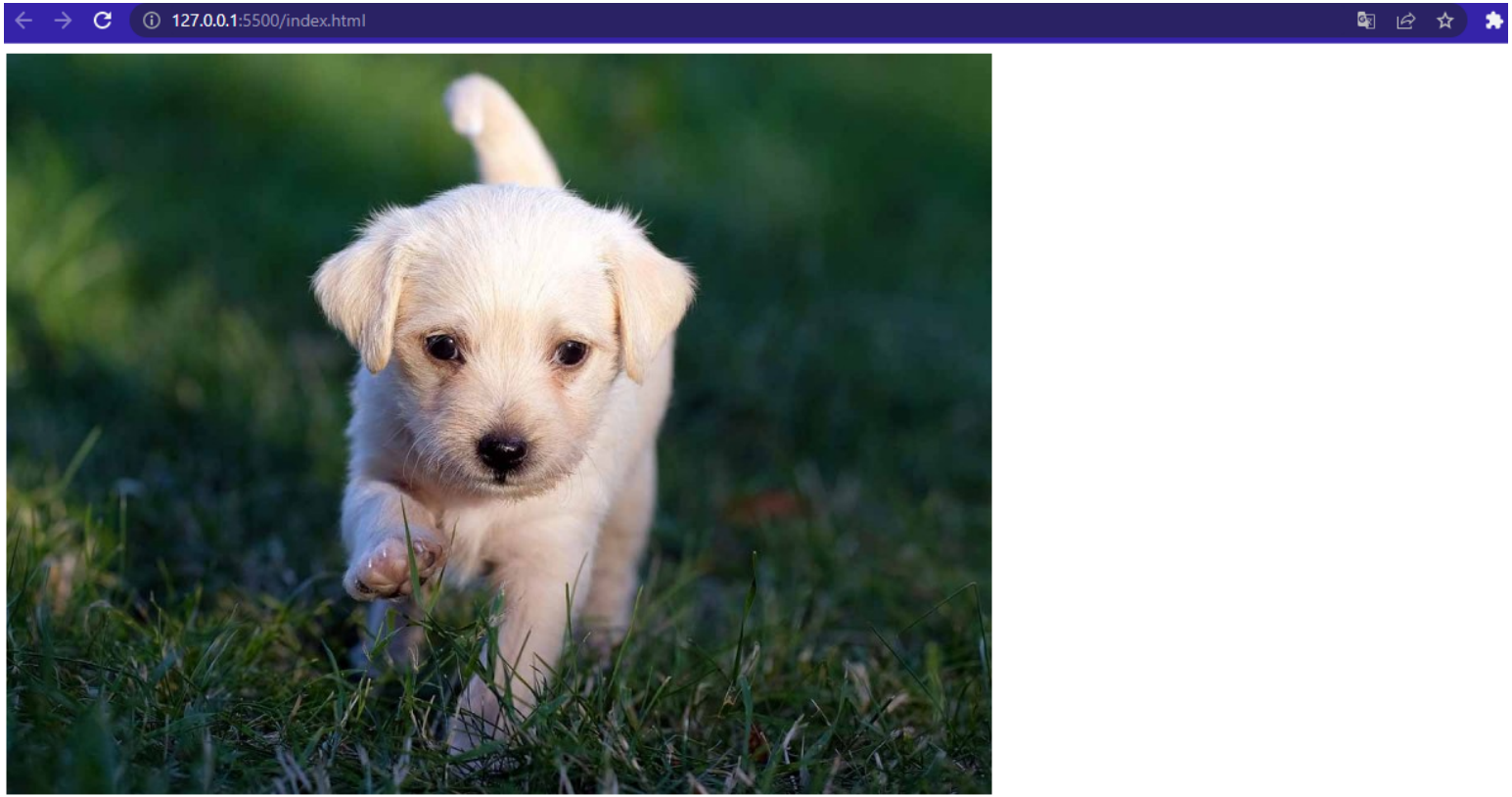
HTML:

```
<div id="container">
    <h1>Bienvenido</h1>
</div>
```

Reemplazo

Con el operador “=” reemplazamos todo el contenido del elemento padre.

```
1   // capturamos el contenedor
2   let contenedor = document.getElementById('container');
3
4   //Acá reemplazamos todo el contenido original
5   contenedor.innerHTML = `
6     <img src='https://cdn2.excelsior.com.mx/media/styles/ir
7   `;
8
9
```


127.0.0.1:5500/index.html

Agregar: Con el operador “+=” agregamos elementos al final del elemento padre.

```
1  // capturamos el contenedor
2  let contenedor = document.getElementById('container');
3
4  //Acá agregamos al final del elemento padre
5  contenedor.innerHTML += `
6    <img src='https://cdn2.excelsior.com.mx/media/styles/ima
7  `;
8
9
```

127.0.0.1:5500/index.html

Bienvenido

Fundamentos JavaScript
Sesión 2 Semana 1