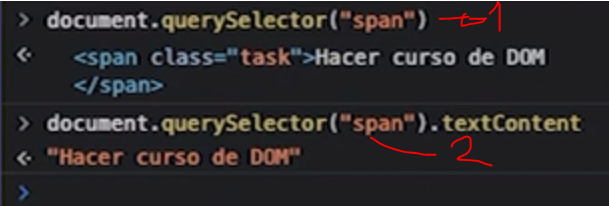


Forma de trabajar con herramienta de desarrollador: usas los métodos de document y cuando colocas la etiqueta, clase o id el sombrea a qué parte te estás refiriendo

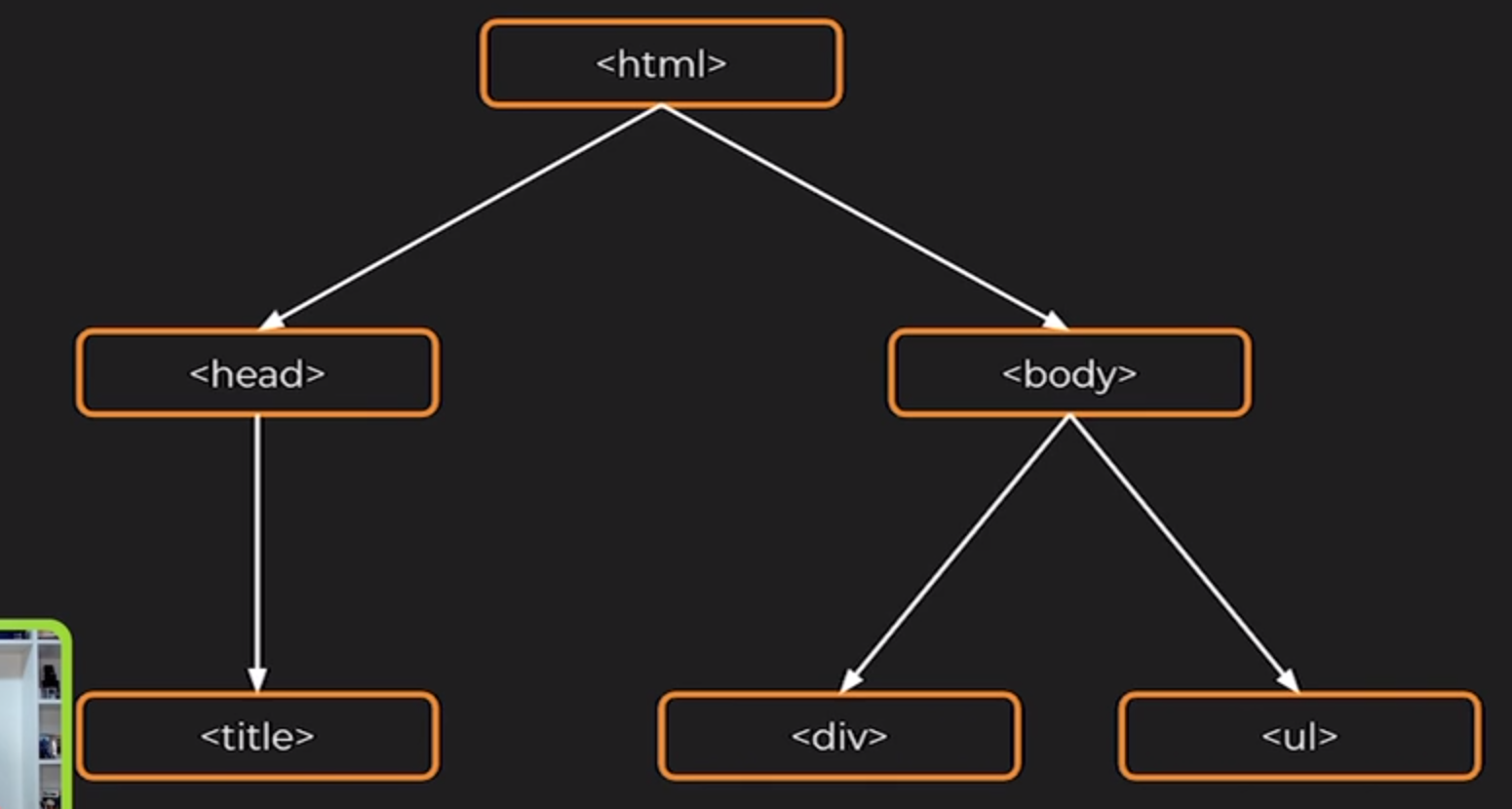
DOM REPRESENTACION DEL OBJETO



querySelector: con cualquier elemento

.textContent muestra el contenido

Este dom sobrevive como objeto en document y en window:



Se genera un árbol de dom, con elementos padre e hijos

Diagrama

Descripción generada automáticamente

Utilizamos el método querySelector para recorrer el árbol del DOM y encontrar el elemento que queremos utilizando *JavaScript*. Pero, existen otro métodos que pueden ser utilizados con la misma finalidad.

* document.getElementById(‘id’) selecciona el elemento por el id
* `document.getElementsByClassName(‘clase’) *retorna un arreglo de los elementos* por el nombre de la clase
* document.getElementsByTagName(‘tag’) *retorna un arreglo de los elementos* por el nombre del tag
* document.querySelectorAll(“selector”) regresa todos los elementos con el mismo nombre

Lo que aprendimos en esta aula:

* Recorrer el árbol del DOM
* Utilizar querySelector para seleccionar los elementos dentro del árbol de DOM
* Funcionamiento de la estructura del DOM

DATA ATRIBUTES: sentencias que se colocan dentro de la etiqueta para uso del formulario en js y no usar su clase : así se nombra en html



Así se nombra en css:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Lo que aprendimos en esta aula:

* Utilizar data-attributes
* Utilizar el método addEventListener para escuchar eventos de un elemento
* Prevenir el comportamiento patrón del formulario

Todos los elementos en nuestro árbol de DOM son nodos y todos los nodos pueden ser accedidos vía JavaScript. Los nodos pueden ser eliminados, creados o modificados. Durante el curso utilizamos el método appendChild que siempre es implementado al final del nodo, para colocar un nodo hijo dentro del nodo padre.

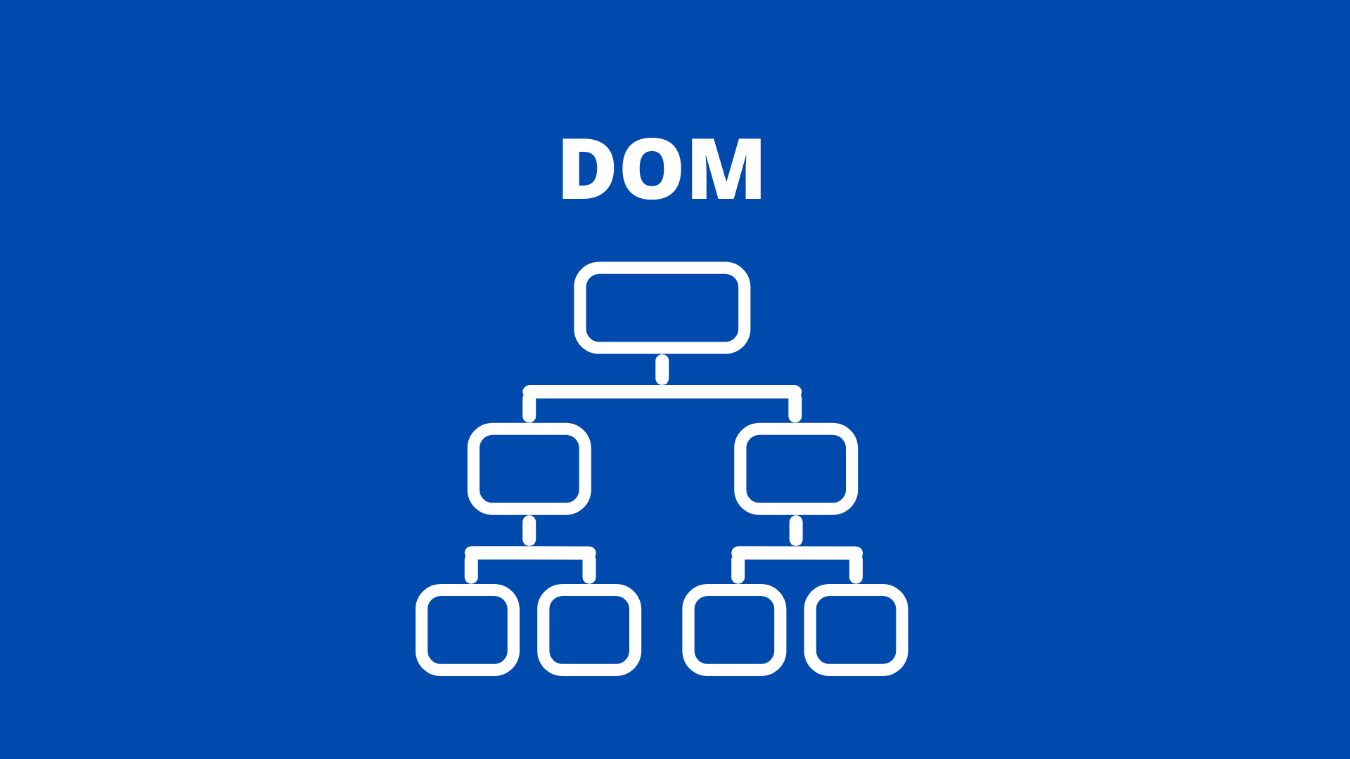
Existen otros métodos que podemos utilizar para manipular nodos:

* insertBefore(padre, hijo): Coloca un nodo antes del otro
* replaceChild(elemento1, elemento2): Sustituye el nodo del elemento 1 por el nodo del elemento 2
* removeChild(elemento): Remueve un nodo del árbol

Qué es DOM?

Ellen Pimentel

28/03/2022



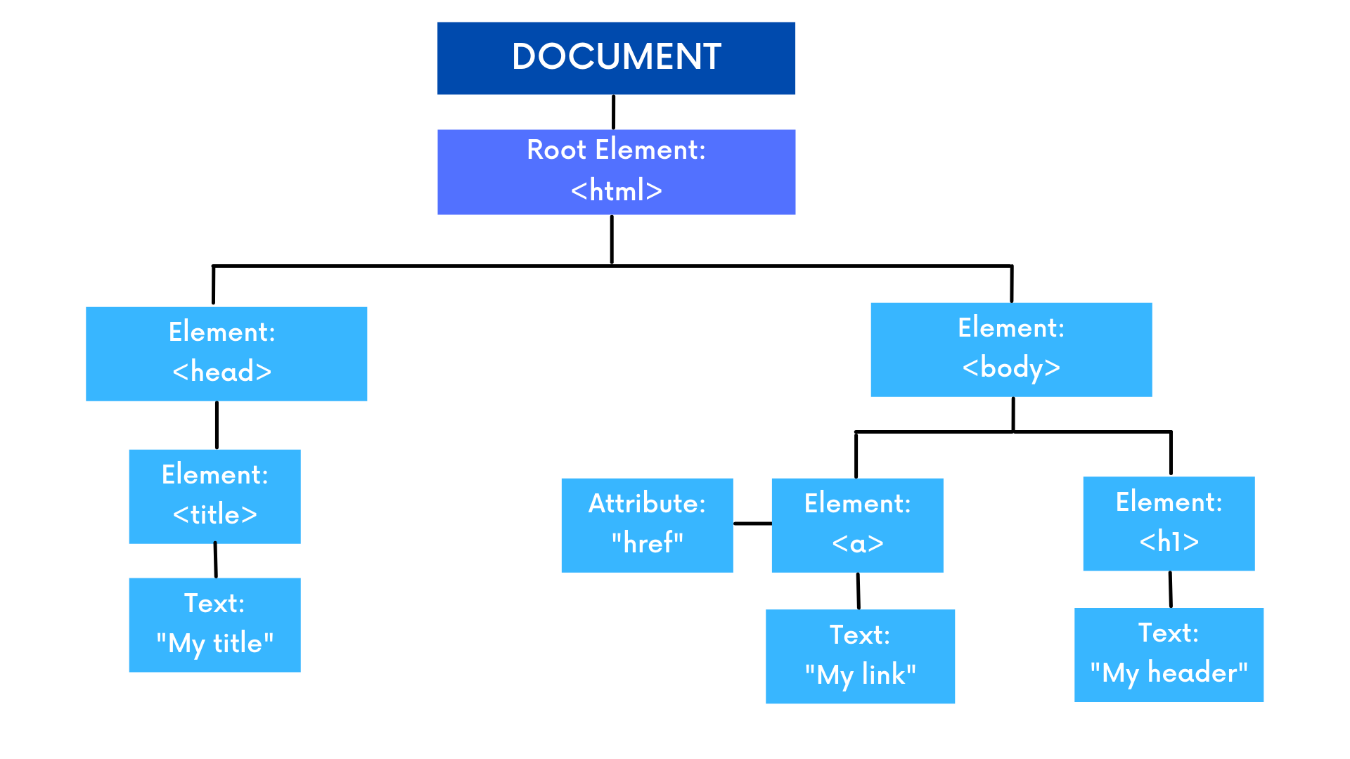
La traducción del acrónimo inglés [DOM](https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction) (Document Object Model) significa Modelo de Documento de Objeto; se trata de modelar todo el HTML.

A partir de su quinta versión, el HTML todavía estático, además de mejoras en la estructura y separación de la semántica de la estilización, ganó interactividad con la creación del lenguaje de programación JavaScript, y fue en este momento que apareció el DOM.

Una de las mejores formas de entender un concepto, y es algo recurrente en el universo de la tecnología, es hacer abstracciones y analogías.

**Estructura**

El DOM es como un árbol genealógico, pero en forma invertida. El elemento que precede al *document* es el *window*, que no es más que la ventana del navegador. En su estructura, *document* se encuentra en la parte superior como un objeto global y su elemento raíz es la etiqueta **html** y todos los demás descienden de él a través de sus ramas (*branches*).

 Fuente: Representación adaptada de W3Schools

La etiqueta **html**, objeto padre, tiene dos objetos hijos: el **head** y el **body** (la cabeza y el cuerpo). Los objetos que siguen a las ramas inferiores se denominan *child*, y los de arriba, *parent*. La etiqueta **head** es *parent* de la etiqueta **title**, y el **body** es *parent* de las etiquetas **a** y **h1**, y así sucesivamente, según la jerarquía. A partir de las etiquetas, derivan los atributos, y de estos, sus valores.

En código, se traduciría así:

<!DOCTYPE html>

<html>

<head>

<title>My title</title>

</head>

<body>

<a href="My link"></a>

<h1>My header</h1>

</body>

</html>

**¿Dónde se inserta?**

Y la pregunta que surge es: ¿pero el DOM es parte de HTML o de JavaScript? De hecho, de ninguno — él es generado por el *browser*. Al cargar la página, el navegador crea el documento, la interfaz, y Javascript usa el DOM para conectarse al HTML.

Para comunicarse entre ellos, es necesario insertar la etiqueta **script** en el archivo HTML, y como buena práctica, debe ser antes del cierre de la etiqueta **body** para que los *scripts* se carguen después del código base.

Puedes hacerlo de dos maneras: escribiendo el código JavaScript dentro de la propia etiqueta script, o insertando la ruta relativa del archivo externo. También como buena práctica, la segunda opción es la más recomendada para la separación de responsabilidades y un mejor mantenimiento del código.

<script>

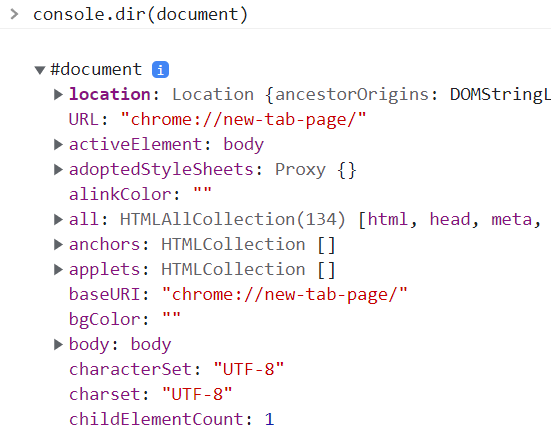
alert("¡Hola, Mundo!")

</script>

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

**Maneras de manejarlo**

En el DevTools del navegador, en la pestaña Console, al escribir **console.dir** (directorio) y pasar el parámetro *document*, se muestra una lista de propiedades y funcionalidades del documento.



Hay varias formas de navegar dentro del DOM, en JavaScript usamos el objeto *document* y a través del punto accedemos a las propiedades y métodos, siendo posible seleccionar, cambiar, borrar y crear elementos a los componentes del sitio *web*, según la estandarización creada por [W3Schools](https://www.w3schools.com/jsref/dom_obj_document.asp). Para realizar estas acciones disponemos de algunos métodos, tales como:

* document.getElementById();
* document.getElementsByClass();
* document.getElementsByTagName();
* document.querySelector();
* document.querySelectorAll();
* document.createElement();
* element.addEventListener();

Con document.querySelector, por ejemplo, entre las opciones que ofrece, podemos cambiar el texto en el documento HTML:

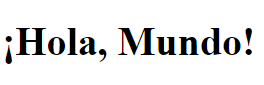
<body>

<h1> ¡Hola! </h1>

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

</body>

document.querySelector("h1").innerText = "¡Hola, Mundo!"



¿Recuerdas que la ventana del navegador está representada por el objeto *window*? Usando otros ejemplos, cuando escribimos la función alert o el método write, es como si lo escribiéramos así:

<script>

window.alert("¡Hola, Mundo!");

window.document.write("¡Hola, Mundo!");

</script>

El *window* se puede omitir, no se requiere en JS, así como con el punto y coma (;) al final de un comando.

<script>

alert("¡Hola, Mundo!")

document.write("¡Hola, Mundo!")

</script>

**Conclusión**

El DOM es un conjunto de objetos y su estructura de datos se representa a través de un diagrama, o en sentido figurado, un árbol de objetos.

Cuando se trata de desarrollo *web* básico, tenemos el trío inseparable: HTML, CSS y JavaScript, cada uno con su propio [rol](https://www.aluracursos.com/blog/html-css-javascript-cuales-son-las-diferencias). Conocer las [bases](https://www.aluracursos.com/blog/algoritmos-y-logica-de-programacion-que-es-eso-y-porque-es-importante) de la programación y lo que ellos representan es fundamental para poder avanzar con *frameworks* y librerías como Angular, Vue.js y React.

La tecnología es un área dinámica, y como en muchos aspectos de nuestra vida, no es necesario tener un don para lograr algo — sino, estudio continuo y mucha práctica