D O M

(Document Object Model)

DEV.F:.

DESARROLLAMOS(PERSONAS);

Objetivo de la sesión

- Hablaremos sobre el DOM en la sesión de hoy.
- Entenderemos qué es el DOM, para qué funciona y cómo manipularlo.
- Haremos algún ejemplo para la clase, y lo que reste de la sesión, seguirán con los ejercicios de funciones (Hacer 4, 5 & 6).

DEV.F

DOM (Document Object Model)

DOM (Document Object Model)

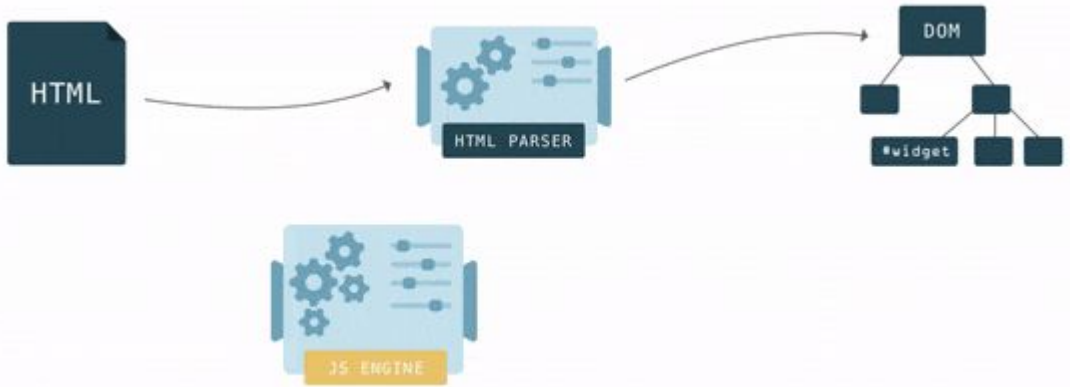
Si estuvieras escuchando música en una aplicación y quisieras pausar u omitir una canción, tendrías que hacerlo a través de la aplicación.

DOM (Document Object Model)

Este proceso es similar a cómo funciona el modelo de objetos de documento o DOM. Aquí, la aplicación de música representa el DOM porque sirve como un medio para realizar cambios en la música.

¿Qué es el DOM?

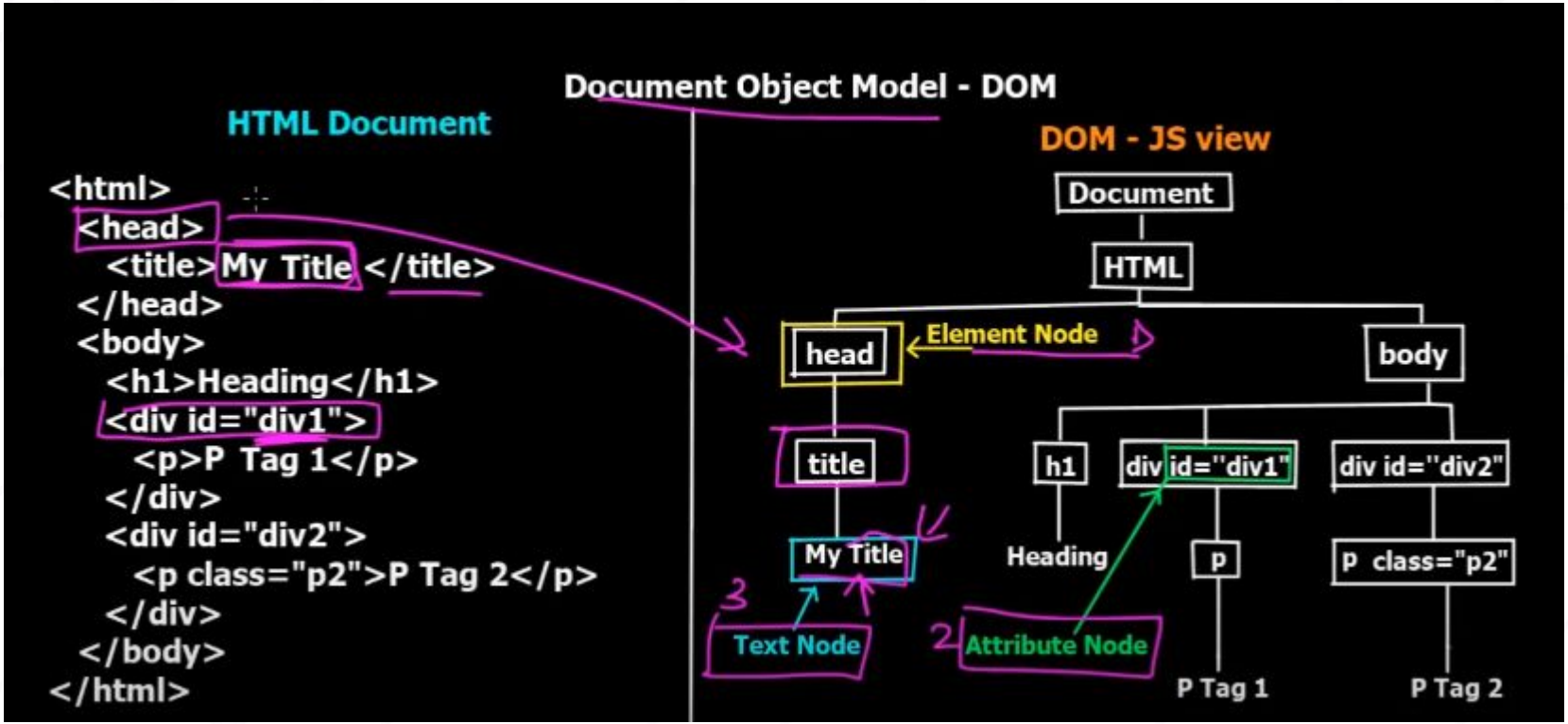
El DOM es una forma que permite a los desarrolladores utilizar la lógica de programación para realizar cambios en su código HTML. Es una forma confiable de realizar cambios que convierten sitios web estáticos en dinámicos.



DOM (Document Object Model)

- El código HTML no se considera parte del DOM hasta que el navegador lo analiza.
- HTML será manipulado a través del DOM (JAVASCRIPT).
- Tendremos el famoso árbol del DOM.

DEV.F


Document Object Model - DOM
HTML Document
<html>
<head>
<title>My Title</title>
</head>
<body>
<h1>Heading</h1>

P Tag 1

P Tag 2

</body>
</html>
DOM - JS view
Document
HTML
head
Element Node
body
title
h1
div id="div1"
div id="div2"
My Title
Heading
p
p class="p2"
3
Text Node
2
Attribute Node
P Tag 1
P Tag 2

NODOS

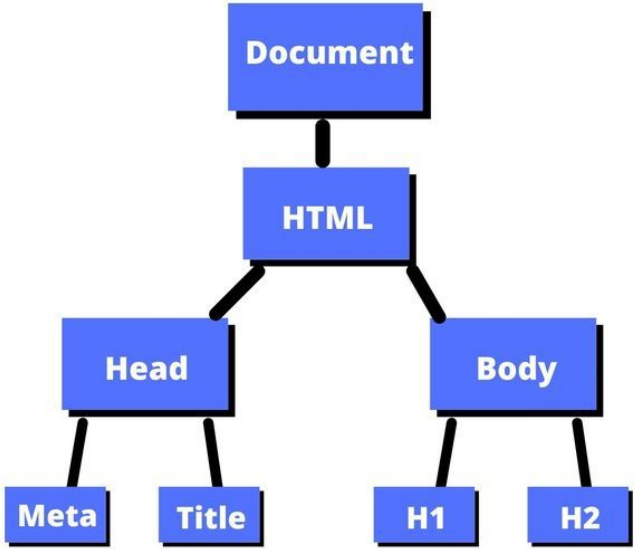
Nuestro documento se llama nodo raíz y contiene un nodo secundario que es el elemento (HTML). El elemento contiene dos hijos que son los elementos y

Ambos elementos tienen hijos propios. <head><body>

DEV.F

NODOS

Aquí hay otra forma de visualizar este árbol de nodos.



IMPLEMENTACIÓN DEL DOM

Todo en el DOM cae en una de estas dos categorías:

- seleccionar elementos.
- manipular elementos.

Después de crear nuestros archivos HTML y CSS,

IMPLEMENTACIÓN DEL DOM

Para obtener acceso a los elementos HTML que deseamos manipular, tenemos que hacer que JavaScript sepa que dichos elementos existen. Esto es lo que generalmente se conoce como "seleccionar" elementos, básicamente vincularlos.

IMPLEMENTACIÓN DEL DOM

En el DOM, no hay una sola forma de localizar y hacer referencia a un elemento para su manipulación. En su lugar, dependerá del selector que haya utilizado en la etiqueta del elemento.

Para ello, asignamos el elemento a una variable. Toma el siguiente formato. Ten en cuenta que todos los selectores DOM están precedidos por el objeto de documento y un punto:

```
> const example = document.[DOMselector]
```

EL OBJETO DOCUMENTO

En Javascript, la forma de acceder al DOM es a través de un objeto llamado document, que representa el árbol DOM de la página o pestaña del navegador donde nos encontramos. En su interior pueden existir varios tipos de elementos, pero principalmente serán ELEMENT o NODE:

- ELEMENT no es más que la representación genérica de una etiqueta: `HTMLElement`.
- NODE es una unidad más básica, la cuál puede ser ELEMENT o un **nodo de texto**.

Todos los **elementos HTML**, dependiendo del elemento que sean, tendrán un tipo de dato específico. Algunos ejemplos:

Tipo de dato	Tipo específico	Etiqueta	Descripción
ELEMENT `HTMLElement`	`HTMLDivElement`	`<div>`	Capa divisoria invisible (en bloque).
ELEMENT `HTMLElement`	`HTMLSpanElement`	`<span>`	Capa divisoria invisible (en línea).
ELEMENT `HTMLElement`	`HTMLImageElement`	`<img>`	Imagen.
ELEMENT `HTMLElement`	`HTMLAudioElement`	`<audio>`	Contenedor de audio.

DOM

En Javascript, cuando nos referimos al DOM hablamos de esta estructura, que podemos modificar de forma dinámica desde Javascript, añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc...

seleccionar elementos del DOM

DEV.F::
DESARROLLAMOS(PERSONAS);

Seleccionar elementos del DOM

Si nos encontramos en nuestro código Javascript y queremos hacer modificaciones en un elemento de la página HTML, lo primero que debemos 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 la clase.

Métodos Tradicionales

Existen varios métodos, los más clásicos y tradicionales para realizar búsquedas de elementos en el documento. Observa que si lo que buscas es un elemento específico, lo mejor sería utilizar getElementById().

Métodos de búsqueda	Descripción
ELEMENT `.getElementById(id)`	Busca el elemento HTML con el id `id`. Si no, devuelve NULL.
ARRAY `.getElementsByClassName(class)`	Busca elementos con la clase `class`. Si no, devuelve `[]`.
ARRAY `.getElementsByName(name)`	Busca elementos con atributo name `name`. Si no, devuelve `[]`.
ARRAY `.getElementsByTagName(tag)`	Busca elementos `tag`. Si no encuentra ninguno, devuelve `[]`.

Métodos Modernos

Método de búsqueda	Descripción
ELEMENT `.querySelector(sel)`	Busca el primer elemento que coincide con el selector CSS `sel`. Si no, NULL.
ARRAY `.querySelectorAll(sel)`	Busca todos los elementos que coinciden con el selector CSS `sel`. Si no, `[]`.

Con estos dos métodos podemos realizar todo lo que hacíamos con los métodos tradicionales mencionados anteriormente e incluso muchas más cosas (*en menos código*), ya que son muy flexibles y potentes gracias a los selectores CSS.

Crear Contenido

Existen una serie de métodos para crear de forma eficiente diferentes elementos HTML o nodos, y que nos pueden convertir en una tarea muy sencilla el crear estructuras dinámicas, mediante bucles o estructuras definidas:

Métodos	Descripción
ELEMENT `.createElement(tag, options)`	Crea y devuelve el elemento HTML definido por el STRING `tag`.
NODE `.createComment(text)`	Crea y devuelve un nodo de comentarios HTML `<!-- text -->`.
NODE `.createTextNode(text)`	Crea y devuelve un nodo HTML con el texto `text`.
NODE `.cloneNode(deep)`	Clona el nodo HTML y devuelve una copia. `deep` es **false** por defecto.
BOOLEAN `.isConnected`	Indica si el nodo HTML está insertado en el documento HTML.

Reemplazar contenido

Comenzaremos por la familia de propiedades siguientes, que enmarcamos dentro de la categoría de reemplazar contenido de elementos HTML. Se trata de una vía rápida con la cuál podemos añadir (*o más bien*, *reemplazar*) el contenido de una etiqueta HTML.

Métodos	Descripción
ELEMENT `.createElement(tag, options)`	Crea y devuelve el elemento HTML definido por el STRING `tag`.
NODE `.createComment(text)`	Crea y devuelve un nodo de comentarios HTML `<!-- text -->`.
NODE `.createTextNode(text)`	Crea y devuelve un nodo HTML con el texto `text`.
NODE `.cloneNode(deep)`	Clona el nodo HTML y devuelve una copia. `deep` es **false** por defecto.
BOOLEAN `.isConnected`	Indica si el nodo HTML está insertado en el documento HTML.

Guías de apoyo

- https://www.freecodecamp.org/news/what-is-the-dom-document-object-model-meaning-in-javascript/
- https://lenguajejs.com/javascript/dom/que-es/
- https://baja.website/manipulacion-del-dom-y-eventos-con-selectores-javascript/