BOOTCAMP

Generación T

streambe

Introducción al DOM (Document Object Model)

streambe

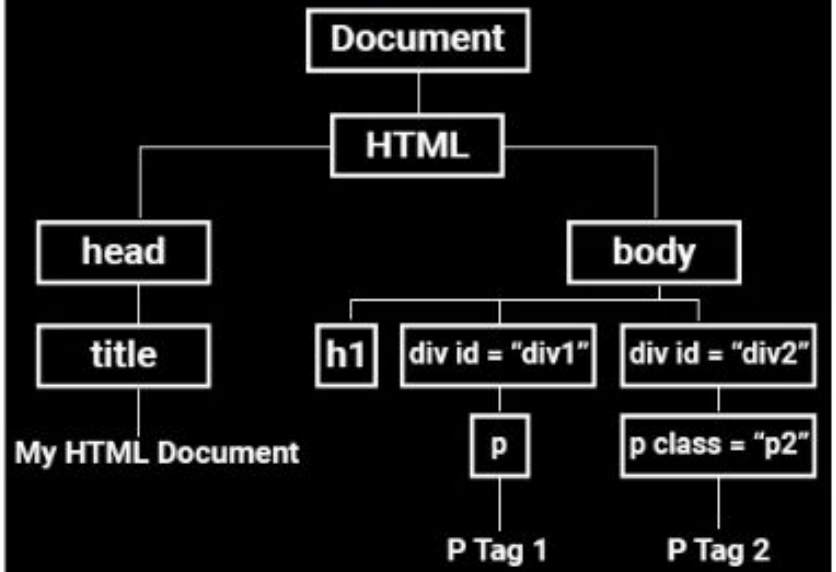
GENERACIÓN T

HTML Document

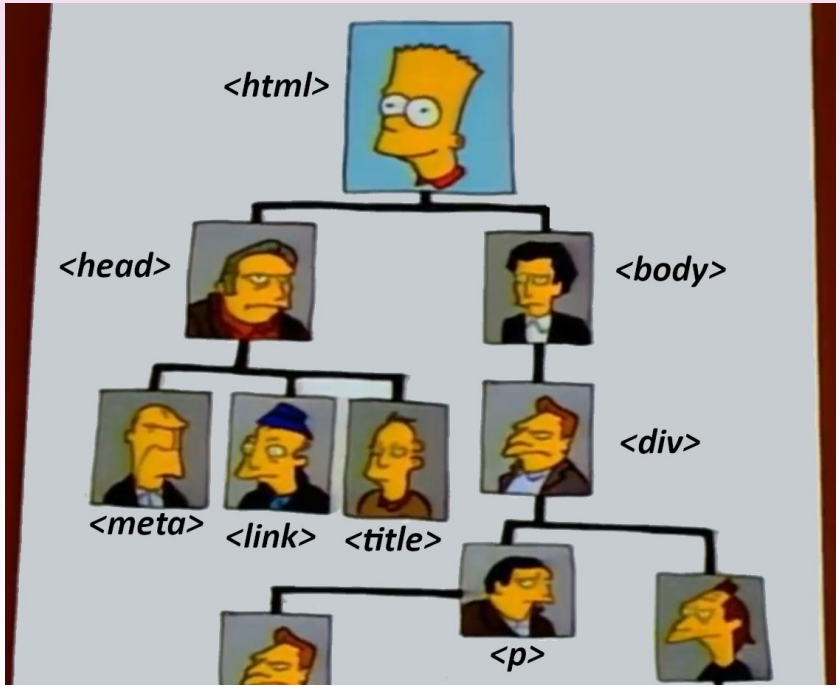
index.html

```
1   <html>
2       <head>
3           <title>My HTML Document</title>
4       </head>
5
6       <body>
7           <h1>Heading</h1>
8           <div id="div1">
9               <p>P Tag 1</p>
10          </div>
11          <div id="div2">
12              <p class="p2">P Tag 2</p>
13          </div>
14      </body>
15  </html>
```

Document Object Model (DOM)



El DOM es un estándar fundamental para el desarrollo web moderno, establecido por el W3C en 1998. Esta presentación explorará cómo este Modelo de Objetos del Documento revolucionó la forma en que interactuamos con páginas web, permitiendo la creación de experiencias dinámicas e interactivas.


<html>
<head>
<body>
<meta>
<link>
<title>

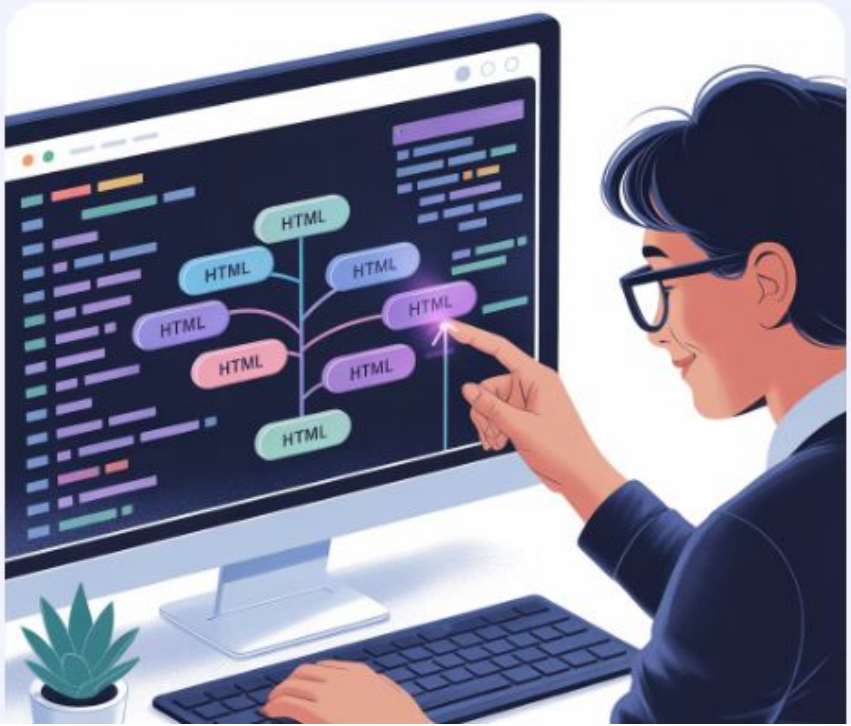
<p>

¿Qué es el DOM?

El DOM (Document Object Model) funciona como una interfaz de programación que permite a los desarrolladores acceder y manipular documentos HTML y XML.

Es esencialmente una representación estructurada del documento web, donde cada elemento se convierte en un objeto manipulable mediante código.

- Convierte páginas web estáticas en interfaces dinámicas
- Permite modificar contenido sin recargar la página
- Facilita la creación de aplicaciones web interactivas



Origen y evolución

1. **1998**

   El W3C establece el DOM Nivel 1, unificando los enfoques incompatibles de Netscape y Microsoft

2. **2000-2004**

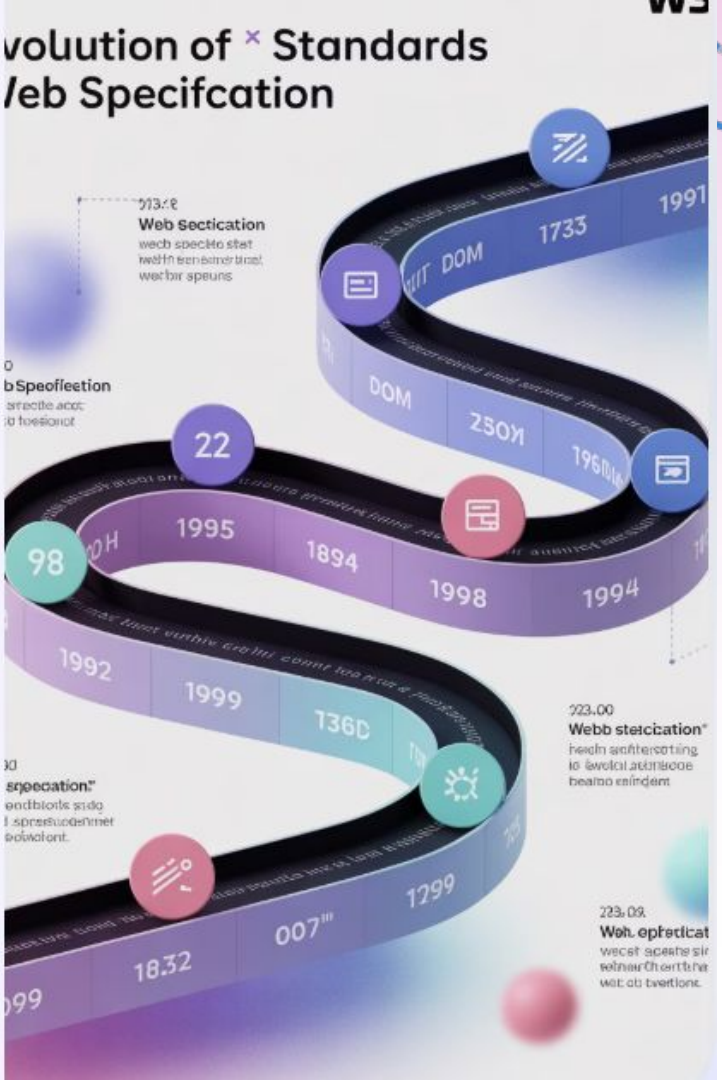
   Desarrollo de DOM Nivel 2 y 3, introduciendo nuevos módulos y funcionalidades

3. **2015**

   Publicación de DOM Nivel 4, integrando mejor con otras tecnologías web modernas

4. **Actualidad**

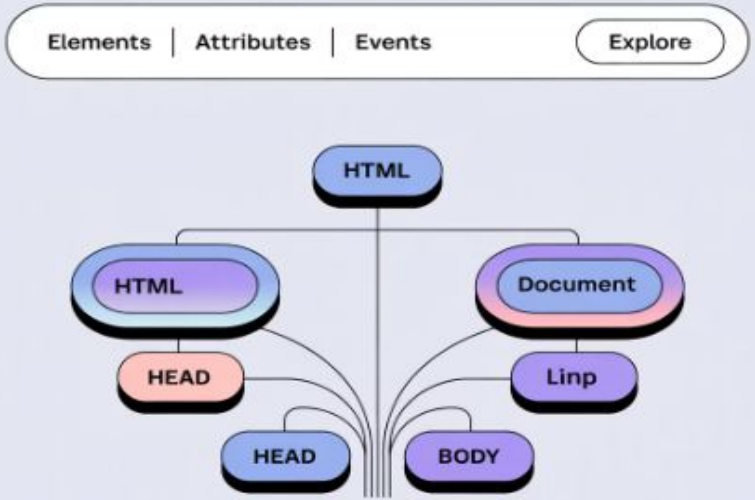
   El DOM es fundamental para frameworks como React, Angular y Vue.js



Estructura básica: El árbol DOM

El DOM representa los documentos HTML como una estructura jerárquica en forma de árbol, donde:

- Cada elemento HTML se convierte en un nodo del árbol
- Los elementos anidados son nodos hijos de sus contenedores
- El documento completo comienza con un nodo raíz
- Las relaciones padre-hijo organizan todos los elementos



¿Cómo Se Manipula El DOM?

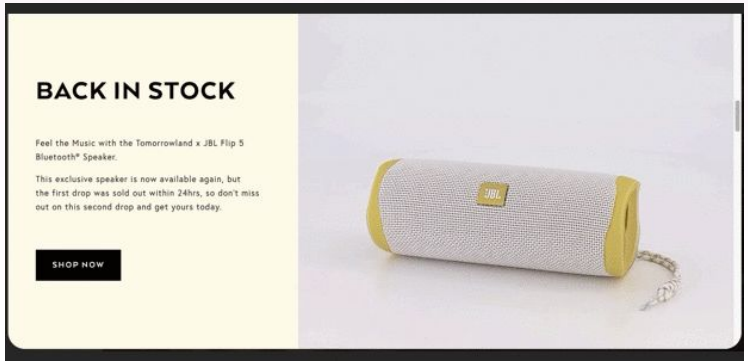
El DOM es la interfaz entre JavaScript y HTML+CSS.

Es decir, cuando cargamos un documento HTML+CSS en el *browser*, el DOM convierte cada *tag* en un Objeto de JavaScript. Esto permite que el *script* manipule los elementos HTML+CSS como si fueran Objetos.

Recapitulando, las páginas *web* previas a JavaScript eran estáticas: carecían de la interacción y el dinamismo que nos brinda JavaScript.

Por ejemplo:

Hoy en día, la misma página web se transforma: ¡Todo se basa en el DOM!



Estructura DOM

Cada etiqueta HTML se transforma en un nodo de tipo "**Elemento**". La conversión se realiza de forma jerárquica.

De esta forma, del **nodo raíz** solamente pueden derivar los nodos **HEAD** y **BODY**.

Cada etiqueta HTML se transforma en un nodo que deriva del correspondiente a su "**etiqueta padre**".

La transformación de las etiquetas HTML habituales genera **dos nodos**

- **Nodo elemento:** correspondiente a la propia etiqueta HTML.
- **Nodo texto:** contiene el texto encerrado por esa etiqueta HTML.

Tipos de nodos en el DOM

Nodos de elemento

Representan las etiquetas HTML como , <p> y <body>. Forman la estructura básica del documento y pueden contener otros nodos.

Nodos de texto

Contienen el texto visible que aparece entre las etiquetas HTML. Son siempre nodos hoja (sin hijos) en el árbol DOM.

Nodos de atributo

Representan las propiedades de los elementos HTML como class, id, href y src que modifican el comportamiento de los elementos.


NO SÉ COMO, PERO DE
UN MOMENTO A OTRO
ESTO SE PONDRÁ
MÁS COMPLEJO
memegenerator.es

Acceso y manipulación con JavaScript

JavaScript es el lenguaje principal para interactuar con el DOM, permitiendo:

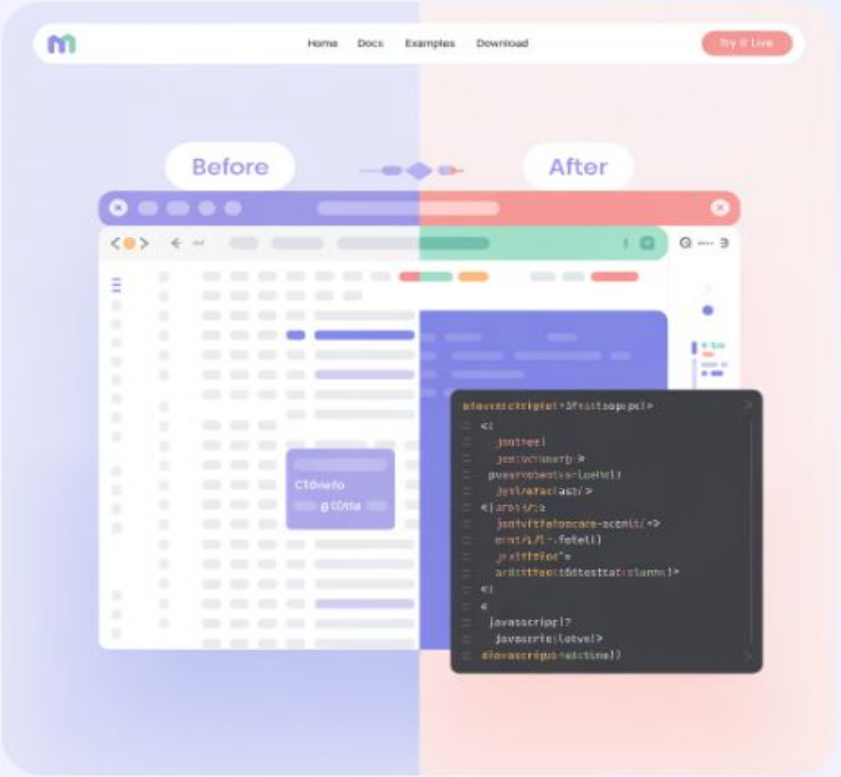
- Seleccionar elementos específicos del documento
- Modificar contenido, estilos y atributos dinámicamente
- Crear o eliminar elementos de la página
- Responder a eventos del usuario como clics o entradas

```
// Ejemplos de métodos DOM
document.getElementById('id')
document.querySelector('.clase')
document.createElement('div')
```

Ejemplo práctico de manipulación DOM

Veamos cómo JavaScript puede interactuar con el DOM para crear una experiencia interactiva:

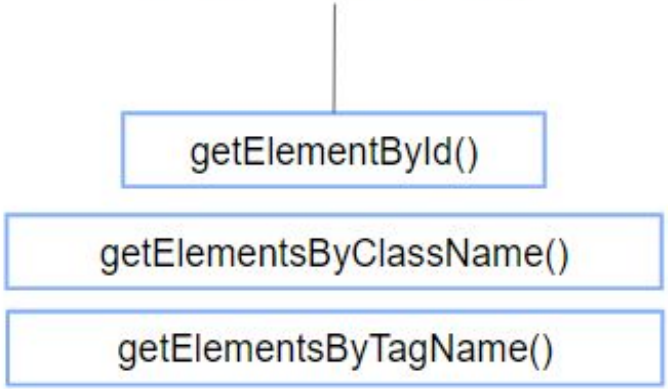
```
// Cambiar texto de un elemento
let titulo =
document.getElementById('titulo');
titulo.textContent = '¡DOM modificado!';

// Crear un nuevo elemento
let nuevoParrafo =
document.createElement('p');
nuevoParrafo.textContent = 'Párrafo creado
dinámicamente';
document.body.appendChild(nuevoParrafo);

// Cambiar estilo
titulo.style.color = 'blue';
```



Acceder a los Nodos

Existen distintos métodos para acceder a los elementos del DOM empleando en la clase Document.

Los más comunes son:



Getelementbyid()

El método **getElementById()** sirve para acceder a un elemento de la estructura HTML, utilizando su atributo ID como identificación.

```
//CODIGO HTML DE REFERENCIA
<div id = "app">
    <p id = "parrafo1" >Hola Mundo</p>
</div>

//CODIGO JS
let div     = document.getElementById("app");
let parrafo = document.getElementById("parrafo1");
console.log(div.innerHTML);
console.log(parrafo.innerHTML);
```

Getelementsbyclassname()

El método **getElementsByClassName()** sirve para acceder a un conjunto de elementos de la estructura HTML, utilizando su atributo class como identificación. Se retornará un Array de elementos con todas las coincidencias:

```
//CODIGO HTML DE REFERENCIA
 <ul>
      <li class="paises">AR</li>
      <li class="paises">CL</li>
     <li class="paises">UY</li>
 </ul>
//CODIGO JS
let paises = document.getElementsByClassName("paises");
console.log(paises[0].innerHTML);
console.log(paises[1].innerHTML);
console.log(paises[2].innerHTML);
```

Getelementsbytagname()

El método **getElementsByTagName()** sirve para acceder a un conjunto de elementos de la estructura HTML, utilizando su nombre de etiqueta como identificación. Esta opción es la menos específica de todas, ya que es muy probable que las etiquetas se repitan en el código HTML.

```
//CODIGO HTML DE REFERENCIA
<div>
      <div>CONTENEDOR 2</div>
      <div>CONTENEDOR 3</div>
</div>
//CODIGO JS
let contenedores = document.getElementsByTagName("div");
console.log(contenedores[0].innerHTML);
console.log(contenedores[1].innerHTML);
console.log(contenedores[2].innerHTML);
```

Ejemplo aplicado: Recorre HTMLcollection con For...Of

```
let paises       = document.getElementsByClassName("paises");
let contenedores = document.getElementsByTagName("div");

for (const pais of paises) {
    console.log(pais.innerHTML);
}

for (const div of contenedores) {
    console.log(div.innerHTML);
}
```

Query Selector

```
<div id="contenedor">
    <p class="texto"></p>
</div>

// puedo seleccionar la etiqueta <p> siguiendo la sintaxis de CSS para selectores:
let parrafo = document.querySelector("#contenedor p")
// seleccionar sólo el contenedor por id con #
let contenedor = document.querySelector("#contenedor")

// o por clase:
parrafo = document.querySelector(".texto")
```

El método **querySelector()** nos permite seleccionar nodos con la misma sintaxis que utilizamos en los selectores de CSS.

Query Selector

Lo interesante del querySelector es que también aplica a pseudo-clases de CSS, brindando un nivel más avanzado de precisión:

```
let radioChecked = document.querySelector(".radio:checked")
```

Suponiendo que tengo elementos html radio button y quiero seleccionar sólo aquel que esté en checked, ésto lo puedo lograr muy fácil con querySelector y la pseudo-clase :checked de CSS.

document.querySelectorAll()

Selección múltiple avanzada

Permite seleccionar todos los elementos que coinciden con un selector CSS específico, devolviendo una NodeList estática.

```
// Seleccionar todos los párrafos
const parrafos =
document.querySelectorAll('p');

// Seleccionar por clase
const items =
document.querySelectorAll('.item');
```

NodeList vs Array

Aunque parece un array, es una colección de nodos que requiere conversión para usar métodos de array.

Iteración necesaria

Para manipular los elementos seleccionados, debemos recorrerlos con un bucle o forEach.

Potente con selectores complejos

Soporta selectores CSS avanzados como '.clase[atributo^="valor"] > span'.

Muchas gracias.

streambe GENERACIÓN T

Nuestras Redes

www.generaciont.org
www.streambe.com
www.instagram.com/generaciont_ar
www.tiktok.com/@generaciont
generaciont@generaciont.org
Cel: 11 61331747

GENERACIÓN T

streambe