DOM

El modelo de objetos de documento (DOM) del W3C es una plataforma y una interfaz de lenguaje neutral que permite que los programas y scripts accedan y actualicen dinámicamente el contenido, la estructura y el estilo de un documento

El DOM HTML representa la forma en que los documentos HTML están estructurados en memoria cuando se

cargan en un navegador web o se manipulan mediante JavaScript.

CARACTERÍSTICAS DEL DOM.

Estructura del Documento HTML

Un documento HTML está compuesto por varios elementos,

como encabezados, párrafos, listas, imágenes, enlaces, formularios y más.

Estos elementos se definen utilizando etiquetas como <h1>, <p>, <ul>, <img>, <a>, <form>, etc.

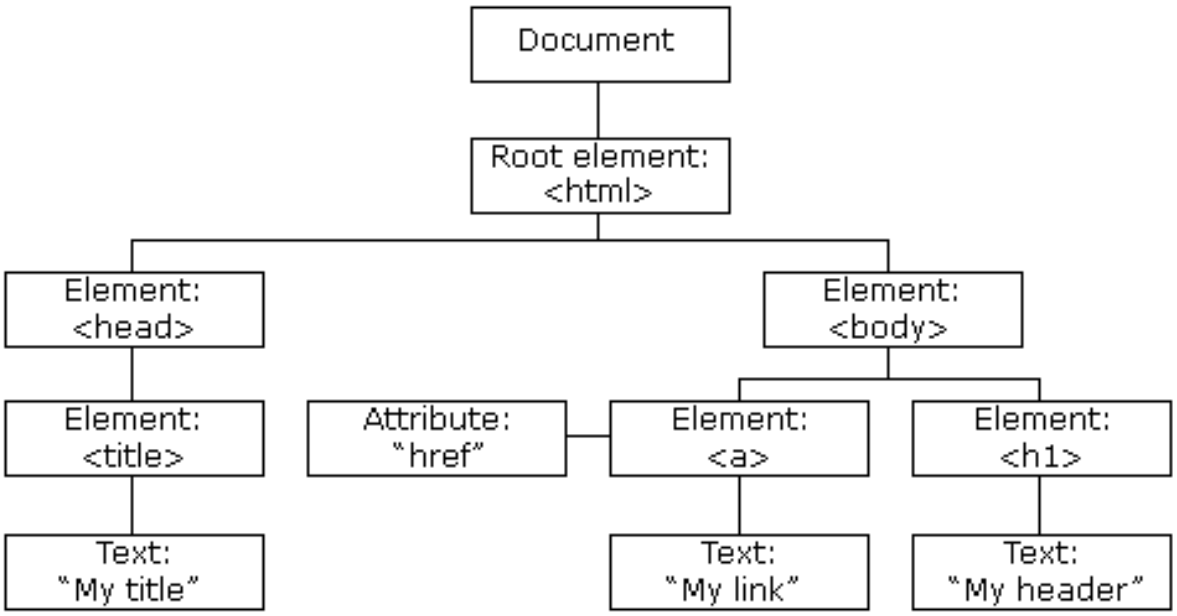
Árbol DOM

Cuando se carga una página web en un navegador, este crea una representación de la

estructura de la página como un árbol de objetos. Cada elemento HTML corresponde a un nodo en el árbol DOM.

Por ejemplo, la raíz del árbol es el elemento <html>, y sus nodos hijos podrían ser <head> y <body>.

Dentro del elemento <body>, tendrás nodos adicionales que representan el contenido y la estructura de la página.



Tipos de Nodos

Los nodos del DOM pueden ser de varios tipos. Los tipos principales relacionados con HTML incluyen:

Nodos de Elemento

Representan elementos HTML y sus atributos.

Nodos de Texto

Representan el contenido de texto dentro de un elemento.

los nodos de texto DOM son elementos esenciales en la representación programática de documentos HTML o XML en el modelo de objetos del documento. Permiten a los desarrolladores acceder y manipular el contenido textual dentro de un documento a través de lenguajes de programación.

Nodos de Atributo

Cada elemento HTML o XML puede tener atributos que proporcionan información adicional sobre el elemento. Estos atributos se representan como nodos en el árbol DOM. Cada nodo de atributo contiene un nombre y un valor.

Nodos de Comentario

Representan los comentarios en el código HTML.

Acceso y Manipulación

Puedes utilizar lenguajes de script como JavaScript para interactuar con el DOM.

Esto te permite modificar dinámicamente el contenido y la estructura de una página web sin necesidad de recargar la página por completo.

Por ejemplo, puedes cambiar texto, agregar o eliminar elementos, actualizar atributos y responder a interacciones de los usuarios.

Eventos

El DOM también maneja eventos, como clics de mouse, pulsaciones de teclas y cambios en el estado de la página. Los eventos pueden ser detectados y manejados mediante JavaScript para ejecutar acciones específicas en respuesta a la interacción del usuario.

JavaScript y el DOM

Con el modelo de objetos con JavaScript es posible manejar y crear HTML dinámico

- JavaScript puede cambiar todos los elementos HTML en la página
- JavaScript puede cambiar todos los atributos HTML en la página
- JavaScript puede cambiar todos los estilos CSS en la página
- JavaScript puede eliminar elementos y atributos HTML existentes
- JavaScript puede agregar nuevos elementos y atributos HTML
- JavaScript puede reaccionar a todos los eventos HTML existentes en la página
- JavaScript puede crear nuevos eventos HTML en la página

Métodos

document.getElementById(id)

Recupera un elemento del DOM por su identificador único y lo devuelve. Es útil para acceder y manipular elementos específicos de la página.

var element = document.getElementById("miElemento");

```
const elementoSumar = document.getElementById('sumar');
const elementoRestar = document.getElementById('restar');
const elementoMultiplicar = document.getElementById('multiplicar');
const elementoDividir = document.getElementById('dividir');
const elementoIgual = document.getElementById('igual');
const elementoC = document.getElementById('c');
const elementosNumeros = document.querySelectorAll('button');
const display = document.getElementById('display');
```

Figure 1 Ejemplo de getElementById

document.querySelector(selector)

Devuelve el primer elemento que coincide con el selector CSS especificado.

var element = document.querySelector(".miClase");

```
const form = document.querySelector('contact-form');
const inputsForm = document.querySelectorAll('input');
```

Figure 2 Ejemplo de querySelector y querySelectorAll

element.innerHTML

Obtiene o establece el contenido HTML dentro de un elemento.

var contenido = element.innerHTML; o element.innerHTML = " Nuevo contenido ";

element.textContent

Obtiene o establece el contenido de texto plano dentro de un elemento.

var texto = element.textContent; o element.textContent = "Nuevo texto";

element.setAttribute(name, value)

Descripción: Establece el valor de un atributo en un elemento.

element.setAttribute("class", "nuevaClase");

element.getAttribute(name)

Obtiene el valor de un atributo en un elemento.

var valor = element.getAttribute("href");

element.classList.add(className)

Agrega una clase CSS al elemento.

element.classList.add("nuevaClase");

element.classList.remove(className)

Elimina una clase CSS del elemento.

element.classList.remove("viejaClase");

element.appendChild(childElement)

Agrega un elemento hijo al final de la lista de hijos del elemento padre.

parentElement.appendChild(childElement);

element.removeChild(childElement)

Elimina un elemento hijo del elemento padre.

parentElement.removeChild(childElement);

element.addEventListener(event, callback)

addEventListener es un método en JavaScript que permite adjuntar un "escuchador" o "listener" a un elemento HTML específico para capturar eventos y responder a ellos. Los eventos son acciones que ocurren en una página web, como hacer clic en un botón, mover el mouse sobre un elemento, presionar una tecla, etc. addEventListener permite ejecutar una función (o un bloque de código) cuando ocurre un evento específico en el elemento objetivo.

```
button1.addEventListener('click', () => {
    parrafo.style.display = 'block';
    button1.disabled = true;
    button2.disabled = false;
})
button2.addEventListener('click', () => {
    parrafo.style.display = 'none';
    button1.disabled = false;
    button2.disabled = true;
})
```

Figure 3 Ejemplo de addEventListener

element.addEventListener("click", miFuncion);

window.onload = function() {}

Ejecuta una función cuando todo el contenido de la página y los recursos externos se han cargado.

Uso: window.onload = function() { /* código a ejecutar */ };