Diplomatura en

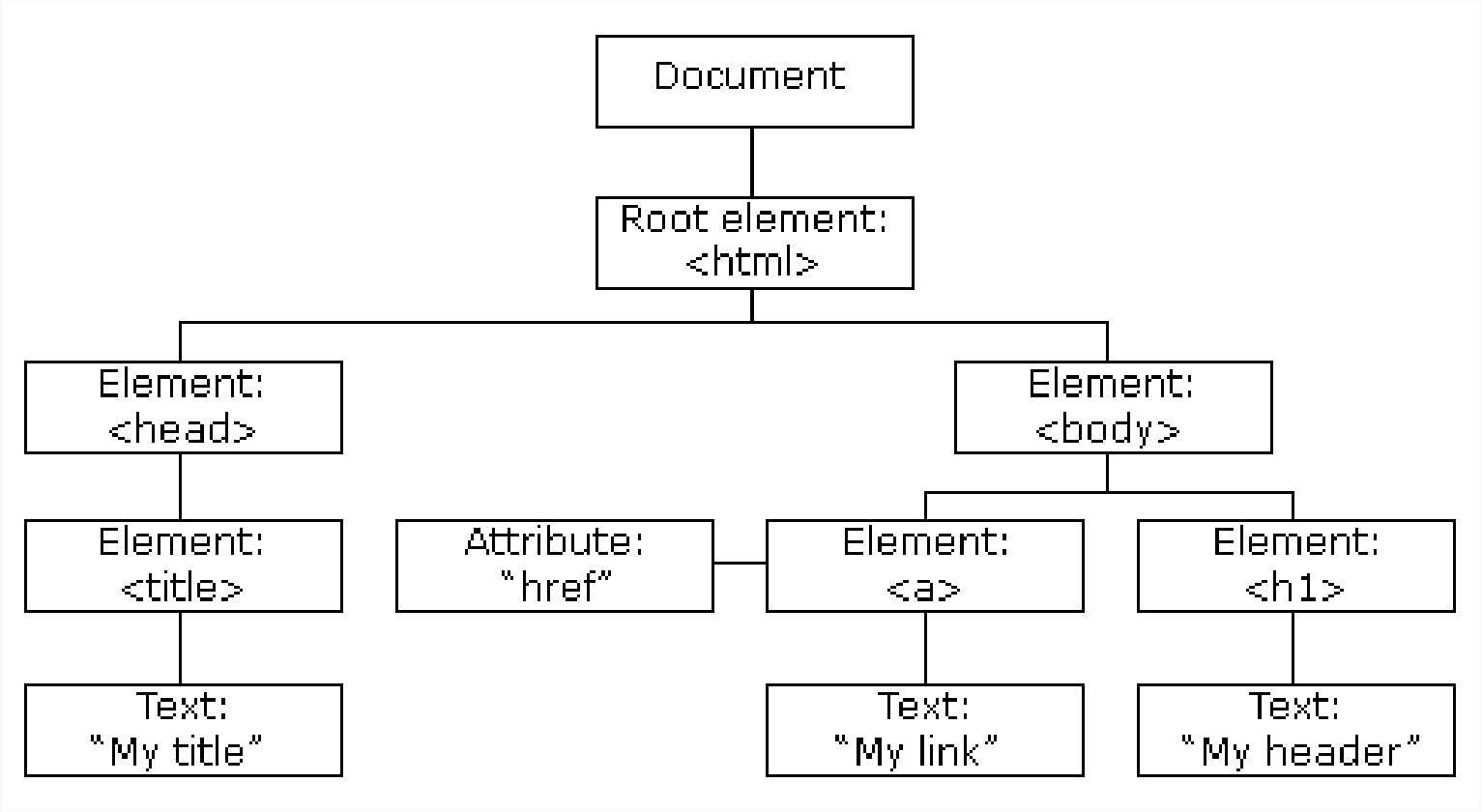
Programación FullStack Developer

GLOBAL ACADEMY

JavaScript
Manipulación del DOM y Eventos

Que es el DOM

- DOM son las siglas de Document Object Model.
- Es una interfaz de programación para documentos HTML y XML.
- Representa la página de manera que los programas puedan cambiar la estructura del documento, estilo y contenido.
- El DOM proporciona una representación del documento como un grupo estructurado de nodos y objetos que tienen propiedades y métodos.
- Permite a JavaScript tener acceso y modificar el contenido de la página web de manera dinámica.
- Es crucial para la interacción del usuario y la interfaz web, permitiendo que las páginas respondan a las acciones del usuario.
- Facilita la creación de aplicaciones web interactivas y dinámicas.



Árbol del DOM

Estructura de árbol

- El DOM se representa como un árbol de nodos.
- Cada página web es un documento que puede ser descrito como un árbol de nodos.
- Cada etiqueta HTML se convierte en un nodo en el árbol del DOM.

Nodos, elementos y atributos

- Nodos: Pueden ser cualquier parte del documento, como un elemento, un atributo o incluso texto.
- Elementos: Son nodos que corresponden a las etiquetas HTML, como , <p>, <a>, etc.
- Atributos: Son propiedades de los nodos que definen características adicionales, como class, id, src.

Visualización

- Incluye un diagrama del árbol del DOM para una página HTML simple.
- Muestra cómo <html> es la raíz, con ramas que incluyen <head> y <body>, y cómo otros elementos como , <p>, y <a> se conectan entre sí.

Acceso al DOM

Métodos de acceso básicos

- document.getElementById(id): Selecciona un elemento por su atributo id. Es muy útil cuando sabes exactamente qué elemento necesitas modificar.
- document.getElementsByClassName(className): Retorna un HTMLCollection de todos los elementos que tienen la clase especificada.
- document.getElementsByTagName(tagName): Obtiene todos los elementos que coinciden con el nombre de etiqueta especificado.

```
// Accede al elemento con id 'header'
let header = document.getElementById('header');
console.log(header);

// Accede a todos los elementos con la clase 'menu-item'
let menuItems = document.getElementsByClassName('menu-item');
console.log(menuItems);

// Accede a todos los elementos 'div'
let divs = document.getElementsByTagName('div');
console.log(divs);
```

Selección moderna de elementos

Métodos modernos de selección

- document.querySelector(selector): Retorna el primer elemento que coincide con el selector CSS especificado. Es muy versátil y útil para acceder a elementos de manera precisa.
- document.querySelectorAll(selector): Retorna todos los elementos dentro del documento que coinciden con el selector CSS especificado. Retorna una NodeList que puede ser recorrida con bucles.

```
// Accede al primer elemento <div> con la clase 'container'
let container = document.querySelector('div.container');
console.log(container);


// Accede a todos los elementos <p> que están dentro de elementos con clase 'text'
let paragraphs = document.querySelectorAll('.text p');
console.log(paragraphs);
```

Manipulación de elementos

Cambiar contenido de elementos

innerText: Modifica o retorna el texto contenido en un elemento, sin incluir etiquetas HTML.

innerHTML: Modifica o retorna el contenido HTML completo dentro de un elemento, permitiendo la inclusión de etiquetas HTML.

```
const heading = document.querySelector('h1');
heading.innerText = 'Bienvenidos a la clase';
heading.innerHTML = '<span>Bienvenidos a la clase</span>';
```

Cambiar estilos de elementos

style: Permite modificar el estilo CSS de un elemento directamente desde JavaScript.

```
const button = document.querySelector('button');
button.style.backgroundColor = 'blue';
button.style.color = 'white';
button.style.padding = '10px 20px’;
button.style.borderRadius = '5px';
```

Manipulación de elementos

Cambiar contenido de elementos

innerText: Modifica o retorna el texto contenido en un elemento, sin incluir etiquetas HTML.

innerHTML: Modifica o retorna el contenido HTML completo dentro de un elemento, permitiendo la inclusión de etiquetas HTML.

```
const heading = document.querySelector('h1');
heading.innerText = 'Bienvenidos a la clase';
heading.innerHTML = '<span>Bienvenidos a la clase</span>';
```

Cambiar estilos de elementos

style: Permite modificar el estilo CSS de un elemento directamente desde JavaScript.

```
const button = document.querySelector('button');
button.style.backgroundColor = 'blue';
button.style.color = 'white';
button.style.padding = '10px 20px’;
button.style.borderRadius = '5px';
```

Creación y eliminación de nodos

Crear elementos

- document.createElement(tagName): Crea un nuevo elemento del tipo especificado.
- append(): Añade nodos o cadenas de texto al final del elemento especificado, permitiendo múltiples nodos y textos

```
const newDiv = document.createElement('div');
newDiv.innerText = 'Este es un nuevo div';
document.body.append(newDiv);  // Uso de append para añadir el div al body
```

Eliminar elementos

removeChild(): Elimina un nodo hijo del elemento especificado. Aunque aún es válido, es menos usado en código moderno.

remove(): Elimina el elemento directamente, sin necesidad de referenciar al padre.

```
// Usando removeChild()
const parent = document.querySelector('#parent');
const child = document.querySelector('#child');
parent.removeChild(child);

// Usando remove()
const elementToRemove =
document.querySelector('#elementToRemove');
elementToRemove.remove();
```

Manipulación de elementos en el DOM

- Estas funciones son esenciales para modificar dinámicamente el contenido de la página.
- Permiten a los desarrolladores añadir o eliminar elementos en respuesta a acciones del usuario, como clics de botón o entradas de formulario.

Manipulación de atributos

Manipulación de atributos de elementos

- getAttribute(attrib): Retorna el valor de un atributo específico del elemento.
- setAttribute(attrib, value): Establece o cambia el valor de un atributo.
- removeAttribute(attrib): Elimina un atributo del elemento.

```
const link = document.querySelector('a');
const hrefValue = link.getAttribute('href');
console.log(hrefValue);  // Muestra el valor del atributo href del enlace

const button = document.querySelector('button');
button.setAttribute('disabled', 'true');
console.log('Botón deshabilitado');

const input = document.querySelector('input');
input.removeAttribute('disabled');
console.log('Input habilitado');
```

Trabajar con clases CSS

Manipulación de clases de elementos

classList: Proporciona métodos para añadir, eliminar, y alternar clases CSS en elementos de una manera sencilla y efectiva.

- classList.add(className): Añade una clase al elemento.
- classList.remove(className): Elimina una clase del elemento.
- classList.toggle(className): Alternar una clase; si el elemento tiene la clase, la elimina, si no, la añade.
- classList.contains(className): Verifica si el elemento tiene una clase específica.

```
const element = document.querySelector('#uniqueElement');
element.classList.add('new-class');
console.log('Clase añadida');
element.classList.remove('old-class');
console.log('Clase eliminada');


const button = document.querySelector('button');
if (!button.classList.contains('active')) {
    button.classList.toggle('active');
    console.log('Clase "active" añadida mediante toggle');
} else {
    button.classList.toggle('active');
    console.log('Clase "active" eliminada mediante toggle');
}


if (element.classList.contains('important')) {
    console.log('El elemento es importante');
}
```

Eventos

Los eventos son acciones o sucesos que ocurren en el navegador que interactúan con una página web, como clicks, presionar teclas, mover el mouse, cargar una página, etc.

JavaScript puede responder a estos eventos a través de funciones denominadas "manejadores de eventos".

Ejemplos de eventos comunes

- Eventos de mouse: click, dblclick, mouseover, mouseout
- Eventos de teclado: keydown, keyup
- Eventos de formulario: submit, change
- Eventos de ventana: load, resize, scroll

Asignación de eventos

Método antiguo: Asignación directa de manejadores de eventos a propiedades de elementos, como element.onclick.

Método moderno: Uso de addEventListener() para manejar eventos, lo cual permite múltiples manejadores para el mismo evento y es más flexible.

Uso de addEventListener()

element.addEventListener('event', handlerFunction);

```
const button = document.querySelector('button');
button.addEventListener('click', function() {
  alert('Botón clickeado!');
});
```

Tipos de eventos

Eventos de mouse

- click: Se dispara cuando el usuario hace clic en un elemento.
- dblclick: Se dispara cuando el usuario hace doble clic en un elemento.
- mouseover: Ocurre cuando el puntero del mouse se mueve sobre un elemento.
- mouseout: Ocurre cuando el puntero del mouse sale del área de un elemento.

Eventos de teclado

- keydown: Se dispara cuando una tecla es presionada.
- keyup: Se dispara cuando una tecla es liberada.
- keypress: (Deprecado) Anteriormente se usaba para capturar la entrada de teclas.

Eventos de formulario

- submit: Se dispara cuando un formulario es enviado.
- change: Ocurre cuando el valor de un elemento de formulario (como un input o select) cambia.
- focus: Se dispara cuando un elemento recibe el foco.
- blur: Se dispara cuando un elemento pierde el foco.

Eventos y el DOM

Acceso y modificación del DOM en manejadores de eventos

Los eventos pueden ser utilizados para modificar el contenido, estructura o estilo del DOM en respuesta a acciones del usuario.

Cambiar el contenido de un elemento cuando se hace clic en él:

```
const paragraph = document.querySelector('p');
paragraph.addEventListener('click', function() {
  this.innerText = 'El texto ha sido cambiado!';
});
```

Alternar estilos de un elemento al pasar el mouse:

```
const box = document.querySelector('.box');
box.addEventListener('mouseover', function() {
  this.style.backgroundColor = 'blue';
});
box.addEventListener('mouseout', function() {
  this.style.backgroundColor = 'red';
});
```

Interacción con formularios

Utilizar eventos para recoger o validar datos de entrada antes de enviar un formulario:

```
const form = document.querySelector('form');
form.addEventListener('submit', function(event) {
  event.preventDefault(); // Evitar el envío automático del formulario
  // Validación o recolección de datos
  console.log('Formulario procesado');
});
```

Dinámica de contenido basado en eventos

Crear interfaces dinámicas que cambien según la interacción del usuario:

```
const select = document.querySelector('select');
select.addEventListener('change', function() {
  const value = this.value;
  document.querySelector('#display').innerText = 'Seleccionaste: ' + value;
});
```

Ejercicios

Tarea: Crear una lista interactiva donde los usuarios puedan agregar y eliminar elementos.

Agregar elementos:
Un campo de entrada de texto y un botón "Agregar".
Al hacer clic en "Agregar", el texto ingresado se añade como un nuevo elemento <li> a una lista <ul>.

Eliminar elementos:
Cada elemento <li> debe tener un botón "Eliminar" que, al hacer clic, elimine el elemento de la lista.

Feedback visual:
Cambiar el color del elemento <li> al pasar el mouse sobre este.

Tarea: Crear un botón que, al hacer clic, cambie el color de fondo de un párrafo.

Requerimientos:

Elementos HTML:
Un párrafo con texto inicial.
Un botón que diga "Cambiar Color".

Interacción JavaScript:
Al hacer clic en el botón, el color de fondo del párrafo debe cambiar a un color aleatorio.