

Tema 3

Manipulación de documentos

Web: Javascript

Lenguajes de Marcas y Sistemas de Gestión de la
Información

ÁRBOL DOM

- El DOM (Document Object Model) es una interfaz de programación que proporciona una representación estructurada de documentos HTML, XHTML o XML.
- Básicamente, convierte un documento web en un modelo de objetos.
- Permite a los desarrolladores acceder, modificar y manipular dinámicamente los elementos y contenido del documento.
- No solo hay DOM en HTML, también en cualquier documento XML.

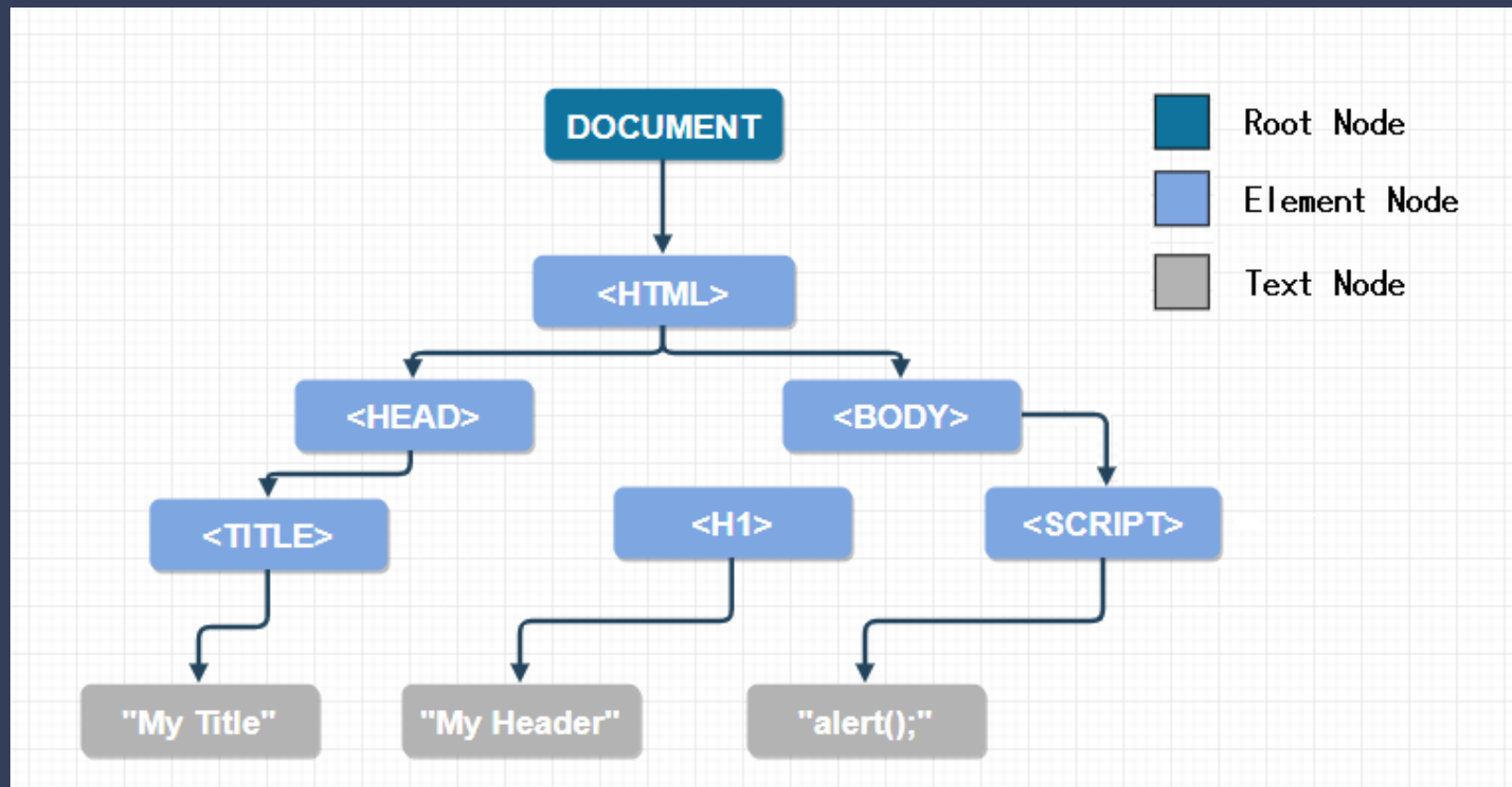
ÁRBOL DOM en HTML

- Una vez que se carga un documento en un navegador, el DOM se construye automáticamente y está disponible para ser accedido y manipulado mediante JavaScript.
- Los desarrolladores pueden acceder a cualquier parte del documento a través del DOM y realizar operaciones como añadir, eliminar o modificar elementos y atributos.

ÁRBOL DOM

- El DOM organiza el contenido del documento en una estructura de árbol, donde cada elemento, atributo y texto se representan como nodos en el árbol.
- Los elementos HTML, como `<div>`, `<p>`, ``, etc., se representan como nodos en el árbol DOM.
- Hay diferentes tipos de nodos.

ÁRBOL DOM



ÁRBOL DOM

- El DOM organiza el contenido del documento en una estructura de árbol, donde cada elemento, atributo y texto se representan como nodos en el árbol.
- Los elementos HTML, como `<div>`, `<p>`, ``, etc., se representan como nodos en el árbol DOM.
- Hay diferentes tipos de nodos.

Tipos de nodos

- **Nodos de Elemento:** Representan elementos HTML y pueden tener nodos secundarios, como hijos, padres y hermanos.
- **Nodos de Texto:** Representan texto dentro de un elemento HTML.
- **Nodos de Atributo:** Representan atributos de elementos HTML.
- **Nodos de Documento:** Representan el documento HTML en sí mismo y son el nodo raíz del árbol DOM.
- **Nodos de Comentarios**

Tipos de nodos

- **Nodos de Elemento:** Representan elementos HTML y pueden tener nodos secundarios, como hijos, padres y hermanos.
- **Nodos de Texto:** Representan texto dentro de un elemento HTML.
- **Nodos de Atributo:** Representan atributos de elementos HTML.
- **Nodos de Documento:** Representan el documento HTML en sí mismo y son el nodo raíz del árbol DOM.
- **Nodos de Comentarios**

Relaciones entre nodos

Padre-Hijo

- Una relación padre-hijo existe cuando un nodo está directamente conectado a otro nodo, siendo el hijo parte del padre.
- Por ejemplo, un elemento `<div>` que contiene un elemento `<p>` tiene una relación padre-hijo.

Relaciones entre nodos

Hermano (Sibling)

- Los nodos que tienen el mismo padre y están en el mismo nivel de jerarquía se denominan hermanos.
- Un nodo puede tener hermanos a la izquierda o a la derecha, dependiendo de su posición en relación con otros nodos del mismo nivel.

Relaciones entre nodos

Ancestro-Descendiente

- Una relación ancestro-descendiente existe cuando un nodo está en un nivel superior de jerarquía y otro nodo está en un nivel inferior, siendo el descendiente parte del ancestro.
- Por ejemplo, un elemento `<body>` puede ser un ancestro de todos los elementos dentro del documento.

Relaciones entre nodos

Texto-Elemento, Atributo-Elemento

- Los nodos de texto representan el contenido de un elemento HTML.
- Los atributos están asociados a elementos y proporcionan información adicional sobre ellos.
- Hay una relación directa entre los nodos de texto y atributo y el elemento al que pertenece, siendo estos hijos del elemento.

Relaciones entre nodos

Ancestro-Descendiente

- Una relación ancestro-descendiente existe cuando un nodo está en un nivel superior de jerarquía y otro nodo está en un nivel inferior, siendo el descendiente parte del ancestro.
- Por ejemplo, un elemento `<body>` puede ser un ancestro de todos los elementos dentro del documento.

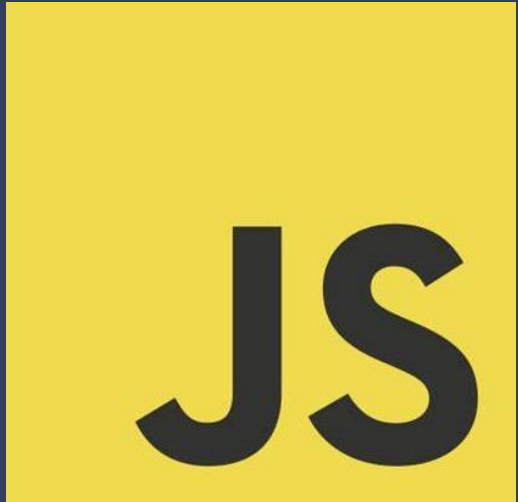
Relaciones entre nodos

Ancestro-Descendiente

- Una relación ancestro-descendiente existe cuando un nodo está en un nivel superior de jerarquía y otro nodo está en un nivel inferior, siendo el descendiente parte del ancestro.
- Por ejemplo, un elemento `<body>` puede ser un ancestro de todos los elementos dentro del documento.

JavaScript

101

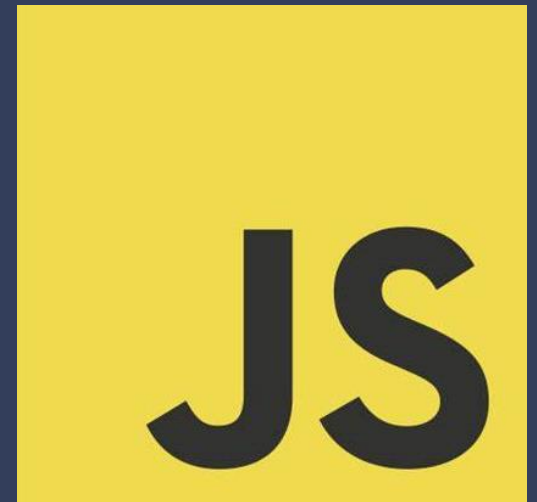
The JavaScript logo, consisting of a yellow square with the letters 'JS' in black, is positioned in the bottom right corner of the slide.

JS

Introducción a Javascript

JavaScript es un lenguaje de programación utilizado principalmente en el desarrollo web para agregar interactividad y funcionalidades dinámicas a las páginas web.

Es un lenguaje de alto nivel, interpretado y multiplataforma.



Introducción a Javascript

Lenguaje de Script: JavaScript es un lenguaje de script, lo que significa que se ejecuta en el navegador del cliente lo que permite la manipulación de elementos de la página web en tiempo real.

Multiparadigma: JavaScript es multiparadigma, lo que significa que soporta programación orientada a objetos, programación funcional y programación imperativa.

Tipado Débil y Dinámico: JavaScript es de tipado débil y dinámico, lo que significa que las variables no están asociadas a un tipo de dato específico y pueden cambiar de tipo durante la ejecución del programa.

Introducción a Javascript

Lenguaje de Script: JavaScript es un lenguaje de script, lo que significa que se ejecuta en el navegador del cliente lo que permite la manipulación de elementos de la página web en tiempo real.

Multiparadigma: JavaScript es multiparadigma, lo que significa que soporta programación orientada a objetos, programación funcional y programación imperativa.

Tipado Débil y Dinámico: JavaScript es de tipado débil y dinámico, lo que significa que las variables no están asociadas a un tipo de dato específico y pueden cambiar de tipo durante la ejecución del programa.

Uso actual

Desarrollo Web: JavaScript se utiliza principalmente para mejorar la experiencia del usuario en páginas web, mediante la manipulación del DOM, validación de formularios, animaciones, etc.

Desarrollo de Aplicaciones Web: Con el surgimiento de frameworks como React, Angular y Vue.js, JavaScript también se utiliza para desarrollar aplicaciones web complejas de una sola página (SPA).

Desarrollo de Servidores: Con Node.js, JavaScript también se puede utilizar para desarrollar aplicaciones de servidor y construir aplicaciones web de lado del servidor.

Sintaxis básica

- **Variables:** let, const, var
- **Tipos de Datos:** number, string, boolean, null, undefined, object, symbol
- **Estructuras de Control:** if, else, switch, for, while
- **Funciones:** Declaración de función, funciones de flecha
- **Objetos:** Creación de objetos, propiedades, métodos
- **Eventos:** Manejo de eventos del DOM

Variables

JavaScript utiliza las palabras clave `let`, `const`, y `var` para declarar variables.

```
let nombre = "Juan";  
const edad = 30;  
var esActivo = true;
```

Imprimir por consola

JavaScript utiliza la función `console.log()` para imprimir datos en la consola del navegador.

```
let nombre = "Juan";  
console.log(nombre);
```

Tipos de datos

JavaScript tiene varios tipos de datos, incluyendo number, string, boolean, null, undefined y object.

```
let numero = 42;  
let texto = "Hola Mundo";  
let esVerdadero = true;  
let nulo = null;  
let indefinido = undefined;  
let objeto = { nombre: "Juan", edad: 30 };
```

Estructuras de control

```
if (edad >= 18) {  
    console.log("Es mayor de edad");  
} else {  
    console.log("Es menor de edad");  
}
```

Estructuras de control

```
if (edad >= 18) {  
    console.log("Es mayor de edad");  
} else {  
    console.log("Es menor de edad");  
}
```

Estructuras de control

```
for (let i = 0; i < 5; i++) {  
    console.log(i);  
}
```

```
let contador = 0;  
while (contador < 5) {  
    console.log(contador);  
    contador++;  
}
```


Funciones

```
function saludar(nombre) {  
    console.log("Hola, " + nombre);  
}
```

```
let sumar = function(a, b) {  
    return a + b;  
};
```

Objetos

```
let persona = {  
  nombre: "Juan",  
  edad: 30,  
  esEstudiante: false,  
  saludar: function() {  
    console.log("Hola, soy " + this.nombre);  
  }  
};
```

```
console.log(persona.nombre); // Acceder a una propiedad  
persona.saludar(); // Llamar a un método
```

Objetos especiales

El objeto document representa el documento HTML cargado en el navegador y proporciona métodos y propiedades para interactuar con él.

```
let titulo = document.querySelector("h1");
```

```
document.write("<h2>Nuevo Título</h2>");
```

Objetos especiales

El objeto window representa la ventana del navegador y proporciona métodos y propiedades relacionadas con la ventana y su contenido.

```
window.alert("¡Hola Mundo!");
```

```
let nombre = window.prompt("Por favor, introduce tu nombre:");  
if (nombre) {  
    console.log("Hola, " + nombre);  
} else {  
    console.log("No se proporcionó un nombre");  
}
```

Objetos de tipo elemento

Los elementos son un tipo especial de nodo que representan elementos HTML en el DOM.

Propiedades y métodos básicos:

- **innerHTML, textContent**: Contenido HTML o texto dentro del elemento.
- **getAttribute(nombre)**: Obtiene el valor de un atributo del elemento.
- **setAttribute(nombre, valor)**: Establece el valor de un atributo del elemento.
- **classList**: Lista de clases CSS aplicadas al elemento.
- **addEventListener(evento, función)**: añade un listener

Objetos de tipo elemento

Propiedades básicas:

- **id**: devuelve el id del elemento.
- **src**: si es una imagen, accede al atributo src.
- **href**: si es un enlace, accede al atributo href.
- **value**: si es un input, accede al atributo value.
- **style**: accede al objeto de estilos CSS
 - `objeto.style.backgroundColor = "red";`
 - `objeto.style.color="green";`
 - `objeto.style.width="100px";`
 - `objeto.style.display="none";`

Objetos de tipo elemento

```
let elemento = document.querySelector("div"); // Obtener un elemento (el primero)
```

```
console.log(elemento.tagName); // Imprimir el nombre de la etiqueta  
console.log(elemento.innerHTML); // Imprimir el HTML interno del elemento  
console.log(elemento.getAttribute("id")); // Obtener el valor del atributo 'id'  
elemento.classList.add("nueva-clase"); // Agregar una nueva clase al elemento
```

```
let elemento = document.querySelector("p#intro"); // Obtener un elemento <p> con  
id intro
```

```
console.log(elemento.textContent); // Imprimir el texto del elemento  
elemento.textContent = "¡Hola Mundo!"; // Cambiar el texto dentro del elemento
```

Obtener elementos

La función `querySelector()` sirve seleccionar un solo elemento del DOM utilizando un selector CSS. Retorna el primer elemento que coincida con el selector especificado. Si no se encuentra ninguna coincidencia, retorna `null`.

```
// Obtener el primer párrafo del documento  
let primerParrafo = document.querySelector("p");
```

```
// Obtener un elemento con un ID específico  
let elementoConId = document.querySelector("#miElemento");
```

```
// Obtener un elemento con una clase específica  
let elementoConClase = document.querySelector(".miClase");
```

Lista de clases

La propiedad `classList` es una propiedad que retorna una colección de las clases CSS de un elemento. Permite agregar, quitar y verificar clases de manera sencilla en un elemento HTML.

```
let elemento = document.getElementById("miElemento");
```

```
// Agregar una clase al elemento  
elemento.classList.add("nuevaClase");
```

```
// Eliminar una clase del elemento  
elemento.classList.remove("claseExistente");
```

```
// Alternar la presencia de una clase  
elemento.classList.toggle("otraClase");
```

Eventos

JavaScript permite manejar eventos del DOM, como clics, cambios, y teclas presionadas. Se deben crear listeners, indicando que hacer cuando el evento ocurra.

```
function avisar() {  
    console.log("El botón fue clickeado");  
}
```

```
let boton = document.querySelector("#miBoton");  
boton.addEventListener("click", avisar );
```

Cambiar clases

```
// Obtener referencia al elemento y al botón
let parrafo = document.getElementById("parrafo");
let boton = document.getElementById("boton");

// Definir función para cambiar la clase
function cambiarClase() {
    parrafo.classList.toggle("resaltado");
}

// Agregar listener al botón
boton.addEventListener("click", cambiarClase);
```

Ejercicios de ejemplo

Crea una página web que contenga un cuadrado de 200px centrado en pantalla y tres botones debajo de él. Cada botón cambiará el color del cuadrado a verde, rojo o azul, respectivamente.

- Crea un archivo HTML con una estructura básica que incluya un `<div>` para el cuadrado y tres `<button>` para los botones.
- Utiliza CSS para dar estilo al cuadrado, centrarlo en pantalla y establecer un tamaño de 200px.
- Utiliza JavaScript para añadir funcionalidad a los botones. Al hacer clic en cada botón, cambia la clase del cuadrado para cambiar su color a verde, rojo o azul.
- Utiliza `classList` para agregar y quitar clases que definan los estilos de los colores.
- Verifica que la página funcione correctamente al hacer clic en cada botón y observar el cambio de color del cuadrado.

Ejercicios de ejemplo

Crea una página web que contenga una galería con 16 imágenes aleatorias. Inicialmente, la galería mostrará solo las primeras 4 imágenes. Cuando se pulse sobre un botón con el texto "Mostrar más", se mostrarán todas las imágenes de la galería.

- Genera una lista de 16 imágenes aleatorias para tu galería. Puedes utilizar imágenes reales o de Picsum.
- Crea un archivo HTML con una estructura básica que incluya un contenedor para la galería de imágenes y un botón con el texto "Mostrar más".
- Utiliza CSS para dar estilo a la galería y asegurarte de que inicialmente solo se muestren las primeras 4 imágenes. Lo más sencilla es agrupando las 4 primeras imágenes de forma independiente y ocultando el resto.
- Utiliza JavaScript para implementar la funcionalidad de mostrar más imágenes. Al hacer clic en el botón "Mostrar más", las restantes imágenes de la galería deben hacerse visibles.
- Verifica que la página funcione correctamente al cargar inicialmente solo las primeras 4 imágenes y al mostrar todas las imágenes al hacer clic en el botón "Mostrar más".

Ejercicios de ejemplo

Un menú off-canvas es un tipo de menú de navegación que no está visible en la pantalla principal de la página web. En su lugar, se encuentra oculto fuera del área visible y se muestra mediante una acción del usuario.

Esto permite liberar espacio en la pantalla principal y proporciona una experiencia de navegación más limpia.

Crea una página web que incluya un menú off-canvas desde el lateral izquierdo que se muestre al pulsar sobre un botón.

1. Utiliza CSS para dar estilo al menú off-canvas y asegurarte de que inicialmente esté oculto fuera del área visible de la pantalla.
2. Utiliza JavaScript para implementar la funcionalidad del menú off-canvas. Al hacer clic en el botón de menú, el menú off-canvas debe deslizarse hacia la pantalla desde el lateral izquierdo.