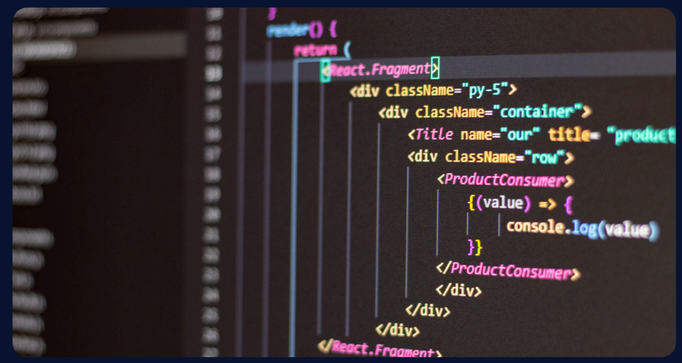


JavaScript

Presented By

Nayfor Martínez
José Mestra

¿Qué es Javascript?

JavaScript (JS) es un lenguaje de programación fundamental para el desarrollo web y, más allá, para una gran variedad de aplicaciones. Es conocido por su capacidad para hacer que las páginas web sean interactivas y dinámicas.

Funciones

Entrada y Salida: Parámetros y return

Las funciones son bloques de código reutilizables para realizar tareas específicas. Puedes definirlas de dos maneras principales:

- Declaración de Función: Se definen con la palabra clave function seguida de un nombre (ej., function miFuncion() {}). La ventaja es que se "elevan" (hoisting), permitiéndote llamarlas antes de su definición en el código.

- Expresión de Función: Se definen asignando una función a una variable (ej., const miFuncion = function() {};). Estas no se elevan, por lo que deben definirse antes de ser llamadas.

Funciones

Entrada y Salida: Parámetros y return

Las funciones interactúan con el resto de tu código a través de:

- Parámetros: Son las variables que una función espera recibir cuando la llamas (ej., function sumar(a, b)). Los valores que le pasas al llamar la función se llaman argumentos.
- Retorno de Valores (return): La palabra clave return se usa para que una función devuelva un valor específico al lugar donde fue llamada. Si no usas return o lo usas sin un valor, la función devuelve undefined por defecto.

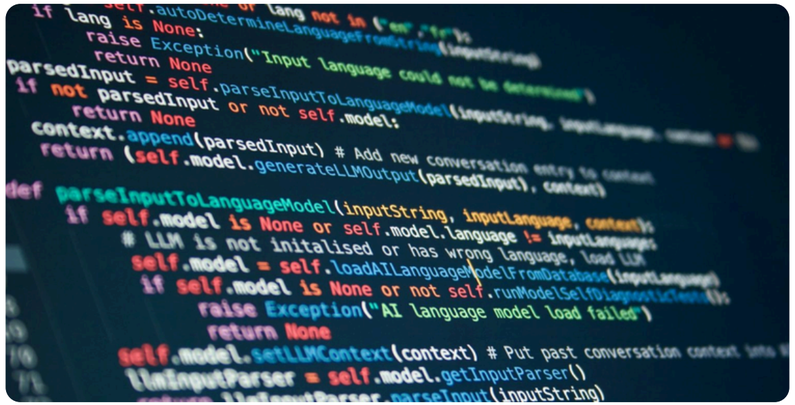
ARRAYS Y SUS METODOS

Un array (arreglo) es una estructura de datos fundamental que permite almacenar varios valores en una sola variable, en lugar de declarar una variable para cada valor.

```
let frutas = ["manzana", "banana", "naranja"];
```

Los valores dentro de un array están organizados de forma ordenada, y cada uno se puede acceder mediante un índice numérico que comienza desde cero (0)

```
console.log(frutas[0]); // "manzana"
console.log(frutas[2]); // "naranja"
```



CREACIÓN DE ARRAYS

Usando corchetes [] (forma más sencilla y común)

```
let frutas = ["manzana", "banana", "naranja"];
```

Usando el constructor Array()

```
let numeros = new Array(1, 2, 3, 4, 5);
```

Array vacío y luego añadir elementos

```
let colores = [];
colores[0] = "rojo";
colores[1] = "verde";
colores[2] = "azul";
```

Manipulación de Arrays

ACCEDER A ELEMENTOS

Cada elemento en un array tiene una posicion, comenzando desde 0

```
let frutas = ['manzana', 'pera', 'uva'];
console.log(frutas[0]); // 'manzana'
console.log(frutas[2]); // 'uva'
```

MODIFICAR ELEMENTOS

Poder cambiar el valor de un elemento directamente

```
frutas[1] = 'banana';
console.log(frutas); // ['manzana', 'banana', 'uva']
```

AGREGAR ELEMENTOS

Al final: **Push()**

```
frutas.push('naranja');
console.log(frutas); // ['manzana', 'banana', 'uva', 'naranja']
```

AGREGAR ELEMENTOS

Al inicio: **Unshift()**

```
frutas.unshift('kiwi');
console.log(frutas); // ['kiwi', 'manzana', 'banana', 'uva', 'naranja']
```

Manipulación de Arrays

ELIMINAR ELEMENTOS

Del final: **Pop()**

```
frutas.pop(); // elimina 'naranja'
console.log(frutas); // ['kiwi', 'manzana', 'banana', 'uva']
```

ELIMINAR ELEMENTOS

Del inicio: **shift()**

```
frutas.shift(); // elimina 'kiwi'
console.log(frutas); // ['manzana', 'banana', 'uva']
```

Metodos Principales de Arrays

Map()

Crea un nuevo array con los resultados de aplicar una función a cada elemento

```
let numeros = [1, 2, 3];
let cuadrados = numeros.map(x => x * x);
console.log(cuadrados); // [1, 4, 9]
```

Filter()

Crea un nuevo array solo con los elementos que cumplan una condición

```
let pares = numeros.filter(x => x % 2 === 0);
console.log(pares); // [2]
```

Find()

Devuelve el primer elemento que cumple con la condición.
Si no encuentra, devuelve **Undefined**

```
let mayorQueUno = numeros.find(x => x > 1);
console.log(mayorQueUno); // 2
```

Metodos Principales de Arrays

forEach()

Ejecuta una función para cada elemento del array, pero no devuelve nada

```
numeros.forEach(num => {
  console.log('Número:', num);
});
```

Sort()

Ordena los elementos como texto por defecto

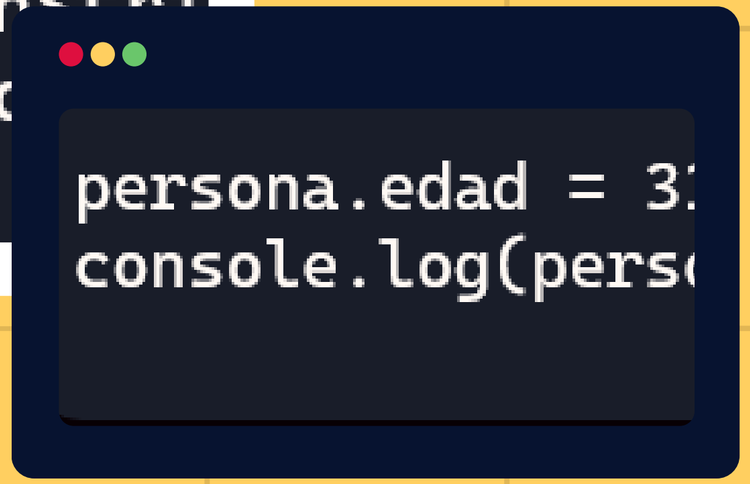
```
let nombres = ['Luis', 'Ana', 'Pedro'];
nombres.sort();
console.log(nombres); // ['Ana', 'Luis', 'Pedro']
```

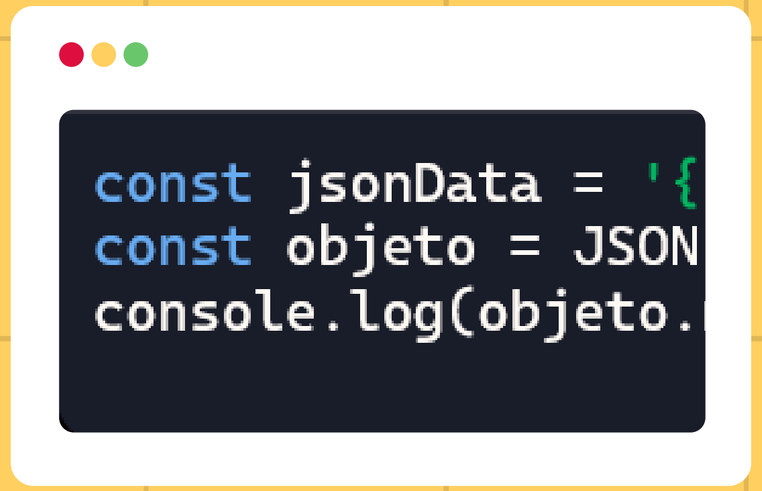
Reverse()

Invierte el orden del array

```
nombres.reverse();
console.log(nombres); // ['Pedro', 'Luis', 'Ana']
```

Objetos y JSON





En JavaScript, los objetos son estructuras que agrupan datos y funciones relacionadas. Se pueden crear mediante la notación literal {}, y acceder a sus propiedades usando . o [], especialmente si los nombres contienen caracteres especiales o son dinámicos. Estos objetos permiten organizar la información y realizar operaciones sobre ella mediante métodos definidos en su interior.

¿Cómo se usan?

Un objeto es una estructura que permite almacenar múltiples valores en forma de propiedades y métodos. Es una manera de organizar datos y funcionalidades en un solo bloque.

JSON (JavaScript Object Notation) es un formato de texto ligero usado para almacenar y transmitir datos, muy utilizado en APIs y bases de datos.

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

```
const auto = new Object();
auto.marca = "Toyota";
auto.modelo = "Corolla";
auto.encender = function() {
  console.log("El auto está ence
};
```

```
{
  "nombre": "María",
  "edad": 28,
  "ciudad": "Bogotá"
}
```

Ejemplos de Uso

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

```
const auto = new Object();
auto.marca = "Toyota";
auto.modelo = "Corolla";
auto.encender = function() {
  console.log("El auto está encendido");
};
```

Conversión entre objetos y JSON

Conversión entre objetos y JSON

1. **Convertir un objeto JavaScript a JSON:**

```
Javascript
```

```javascript
const jsonString = JSON.stringify(persona);
console.log(jsonString); // {"nombre":"Juan","edad":30}
```

2. **Convertir JSON a un objeto JavaScript:**

```
Javascript
```

```javascript
const jsonData = '{"nombre": "Carlos", "edad": 25}';
const objeto = JSON.parse(jsonData);
console.log(objeto.nombre); // Carlos
```

¿ QUE ES EL DOM?

El DOM (Document Object Model) es una representación estructurada del documento HTML, que el navegador crea cuando carga una página web. Permite a JavaScript acceder y manipular los elementos del documento, como cambiar el contenido, estructura o estilo de una página en tiempo real.

El DOM convierte cada parte del HTML (etiquetas, texto, atributos) en un nodo dentro de una estructura de árbol. Por ejemplo, si tenemos este HTML

```
<html>
  <body>
    <h1>Hola Mundo</h1>
  </body>
</html>
```

El DOM representaría esto como un árbol de nodos: document > html > body > h1.

JavaScript accede a este árbol con objetos como document, document.getElementById(), document.querySelector(), etc

Selección y Manipulación de Elementos

SELECCIONAR ELEMENTOS

Seleccionar por ID

```
let parrafo = document.getElementById('mensaje');
```

SELECCIONAR ELEMENTOS

Seleccionar por clase o etiqueta

```
let botones = document.getElementsByClassName('btn');
let divs = document.getElementsByTagName('div');
```

Manipular Contenido y Estilos

```
let titulo = document.getElementById('titulo');

// Cambiar el texto
titulo.textContent = 'Nuevo Título';

// Cambiar el HTML interno
titulo.innerHTML = '<em>Título con énfasis</em>';

// Cambiar estilos
titulo.style.color = 'red';
titulo.style.fontSize = '24px';
```

Crear, Agregar y Eliminar Elementos

```
let nuevoParrafo = document.createElement('p');
nuevoParrafo.textContent = 'Este es un párrafo nuevo';

document.body.appendChild(nuevoParrafo); // lo agrega al final del body

// Eliminar un elemento
nuevoParrafo.remove(); // elimina el párrafo creado
```

EVENTOS

Los eventos del DOM son acciones que ocurren en la página, como cuando el usuario hace clic o presiona una tecla.

EJEMPLO
Cuando se hace clic en un elemento

Tipo de Evento: Click.
element.addEventListener('click', fn)

```
<button id="miBoton">Haz clic</button>


<script>
  const boton = document.getElementById('miBoton');


  boton.addEventListener('click', () => {
    alert('¡Hiciste clic!');
  });
</script>
```

PRINCIPALES EVENTOS

- **MOUSEOVER / MOUSEOUT:** Al pasar o quitar el mouse de un elemento
- **KEYDOWN / KEYUP:** Al presionar o soltar una tecla
- **SUBMIT:** Cuando se envía un formulario
- **CHANGE:** Cuando cambia el valor de un input
- **IMPUT:** Cuando el usuario escribe en un campo
- **LOAD:** Cuando se termina de cargar la página o imagen
- **DOMCONTENTLOADED:** Cuando el HTML se ha cargado por completo
- **RESIZE:** Cuando se cambia el tamaño de la ventana
- **SCROLL:** Cuando el usuario hace scroll en la página

Asincronía básica

Asincronía en JavaScript: JavaScript es un lenguaje single-threaded, lo que significa que ejecuta una tarea a la vez. Sin embargo, para manejar operaciones que toman tiempo (como consultas a bases de datos o peticiones a servidores), se usa la programación asincrónica.

Promesas
Las promesas son una alternativa más moderna y legible para manejar la asincronía en JavaScript. Una promesa tiene 3 estados:

1. Pendiente (pending): cuando la operación aún no ha finalizado.
2. Resuelta (fulfilled): cuando la operación se completa exitosamente.
3. Rechazada (rejected): cuando ocurre un error.

Callbacks:

Un callback es una función que se pasa como argumento a otra función y se ejecuta después de que la primera termine su proceso. Es una forma tradicional de manejar la asincronía en JavaScript.

Ejemplo de Callbacks

Javascript

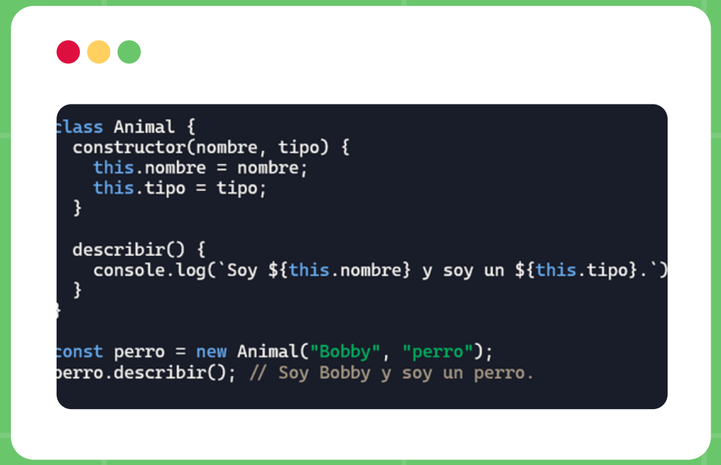
```
function obtenerUsuario(id, callback) {
  setTimeout(() ⇒ {
    const usuario = { id, nombre: "Carlos" };
    callback(usuario);
  }, 2000); // Simula una demora de 2 segundos
}

obtenerUsuario(1, (usuario) ⇒ {
  console.log(`Usuario obtenido: ${usuario.nombre}`);
});
```

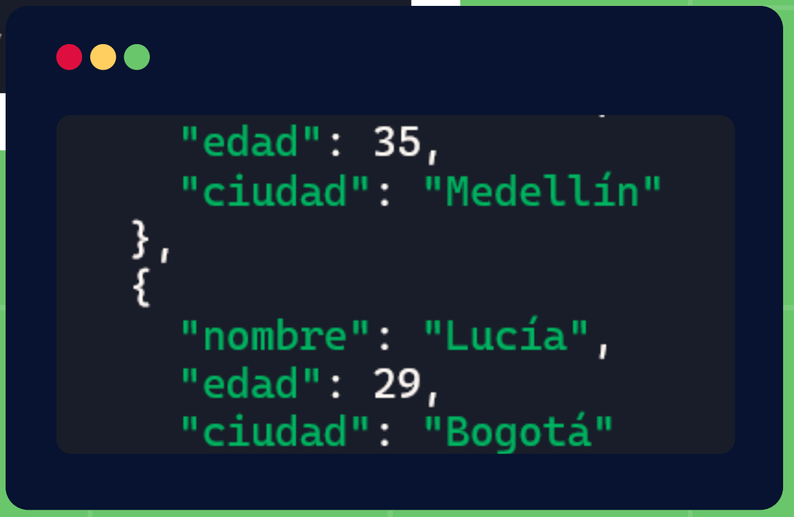
Ejemplo de Promesa básica

```
function buscarUsuario(id) {
  return new Promise((resolve, reject) ⇒ {
    setTimeout(() ⇒ {
      if (id === 1) {
        resolve({ id: 1, nombre: "Laura" });
      } else {
        reject("Usuario no encontrado");
      }
    }, 2000);
  });
}

buscarUsuario(1)
  .then((usuario) ⇒ console.log(`Usuario encontrado: ${usuario.nombre}`))
  .catch((error) ⇒ console.log("Error:", error));
```


class Animal {
constructor(nombre, tipo) {
this.nombre = nombre;
this.tipo = tipo;
}
describir() {
console.log(`Soy ${this.nombre} y soy un ${this.tipo}.`)
}
}
const perro = new Animal("Bobby", "perro");
perro.describir(); // Soy Bobby y soy un perro.


const usuario = {
nombre: "Ana",
edad: 25
};
// Agregar un nuevo método dinámicamente
usuario.saludar = function() {
console.log(`Hola, mi nombre es ${this.nombre}.`);
};
usuario.saludar(); //


"edad": 35,
"ciudad": "Medellín"
},
{
"nombre": "Lucía",
"edad": 29,
"ciudad": "Bogotá"

GRACIAS