EVIDENCIAS 7

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Curso JavaScript - Desarrollo Personal</title>
    <style>
        body {
            font-family: 'Roboto', sans-serif; /* Fuente moderna y limpia */
            margin: 20px;
            background-color: #f0f2f5; /* Fondo gris claro y suave */
            color: #333; /* Texto principal oscuro */
        }
        h1 {
            color: #2c3e50; /* Azul oscuro para el título principal */
            text-align: center;
            font-weight: 700; /* Negrita */
            margin-bottom: 30px;
        }
        h2 {
            color: #34495e; /* Azul grisáceo para los subtítulos */
            border-bottom: 2px solid #3498db; /* Línea azul para destacar */
            padding-bottom: 8px;
            margin-top: 35px;
            font-weight: 600;
        }
        h3 {
            color: #3498db; /* Azul medio para los encabezados de sección */
            margin-top: 25px;
            font-weight: 500;
        }
        section {
            background-color: #ffffff; /* Fondo blanco para las secciones */
            margin-bottom: 25px;
            padding: 20px 25px;
            border-radius: 10px; /* Bordes más suaves */
            box-shadow: 0 4px 10px rgba(0,0,0,0.08); /* Sombra más pronunciada pero sutil */
```

```
Lín. 25, col. 30  Espacios: 4  UT
```

```javascript
// main.js - Código del curso JavaScript.info

// TEMA: 6.1 Recursión y pila — 24/10/2025
// Demuestra recursión con factorial y suma de números.
function factorial(n) {
    if (n === 0 || n === 1) return 1;
    return n * factorial(n - 1);
}
function sumNumbers(n) {
    if (n === 1) return 1;
    return n + sumNumbers(n - 1);
}
const numFactorial = 5, resFactorial = factorial(numFactorial);
const numSum = 4, resSum = sumNumbers(numSum);
console.log(`Factorial de ${numFactorial}: ${resFactorial}`);
console.log(`Suma hasta ${numSum}: ${resSum}`);
document.getElementById('output-6-1').innerHTML = `Factorial de ${numFactorial}: ${resFactorial

// TEMA: 6.2 Parámetros de descanso y sintaxis de propagación — 24/10/2025
// Explora rest parameters (...args) y spread syntax (...array).
function showNames(firstName, lastName, ...titles) {
    return `Hola, ${firstName} ${lastName}, tus títulos son: ${titles.join(', ')}`;
}
const showNamesResult = showNames("Julio", "Verne",  let arr1: number[]  a");
let arr1 = [1, 2], arr2 = [3, 4], combinedArr = [...arr1, ...arr2, 5];
function multiply(a, b, c) { return a * b * c; }
let numbersToMultiply = [2, 3, 4], multiplyResult = multiply(...numbersToMultiply);
console.log(showNamesResult);
console.log("Array combinado:", combinedArr);
console.log("Multiplicación con spread:", multiplyResult);
document.getElementById('output-6-2').innerHTML = `Rest: ${showNamesResult}<br>Spread (arrays):

// TEMA: 6.3 Alcance variable, cierre — 24/10/2025
// Alcance (global, función, bloque) y closures (funciones que recuerdan su entorno).
let globalVar = "Global";
function outerFunction() {
```

piedades del objeto 7 > JS main.js > ...

```javascript
// main.js

// TEMA: 7.1 Banderas y descriptores de propiedad
// Explora las banderas 'writable', 'enumerable' y 'configurable' de las propiedades de objeto.

// Objeto 'user' y su descriptor de 'name' por defecto (todas las banderas en true).
const user = { name: "Sofia" };
const descriptorName = Object.getOwnPropertyDescriptor(user, 'name');
console.log("Descriptor 'user.name' por defecto:", descriptorName);

// 1. writable: false (no se puede cambiar el valor)
Object.defineProperty(user, 'name', { writable: false });
user.name = "Pedro"; // Ignorado o TypeError en 'use strict'.
console.log("user.name (writable: false):", user.name);

// 2. enumerable: false (no aparece en bucles como for...in)
const car = { brand: "Toyota", model: "Corolla" };
Object.defineProperty(car, 'brand', { enumerable: false });
let carProperties = [];
for (let key in car) { carProperties.push(key); }
console.log("Propiedades de 'car' (brand no enumerable):", carProperties);

// 3. configurable: false (no se puede borrar ni cambiar sus banderas)
const settings = { version: "1.0" };
Object.defineProperty(settings, 'version', { configurable: false });
delete settings.version; // Fallará.
console.log("settings.version (configurable: false):", settings.version);

// Inyectar resultados en el HTML para el tema 7.1
document.getElementById('output-7-1').innerHTML = `
    <p><strong>Descriptor 'user.name' por defecto:</strong> <pre>${JSON.stringify(descriptorName, null,
    <p><strong>'user.name' (writable: false):</strong> ${user.name}</p>
    <p><strong>Propiedades de 'car' (brand no enumerable):</strong> ${carProperties.join(', ')}</p>
    <p><strong>'settings.version' (configurable: false):</strong> ${settings.version}</p>
    <p class="console-output">Mira la consola (F12) para más detalles.</p>
`;
```

```javascript
// TEMA: 7.2 Captadores y configuradores de propiedades (Getters y Setters)
// Define propiedades con lógica personalizada al leer (getter) o escribir (setter).

// Objeto 'person' con getter y setter para 'fullName'.
const person = {
    firstName: "linda",
    lastName: "sofia",
    get fullName() { return `${this.firstName} ${this.lastName}`; },
    set fullName(value) { [this.firstName, this.lastName] = value.split(' '); }
};

// Acceder (llama al getter) y asignar (llama al setter).
console.log("Nombre completo inicial (getter):", person.fullName);
person.fullName = "Carlos Ruiz";
console.log("Nuevo nombre completo (setter):", person.fullName);

// Ejemplo práctico: 'age' con validación en el setter.
const userWithAge = {
    _age: 0,
    set age(value) { if (value < 0) console.error("Edad no puede ser negativa."); else this._age = value
    get age() { return this._age; }
};

// Asignar edad válida e inválida.
userWithAge.age = 15;
console.log("Edad válida:", userWithAge.age);
userWithAge.age = -5; // Error en consola.
console.log("Edad después de intentar inválida:", userWithAge.age);

// Inyectar resultados en el HTML para el tema 7.2
document.getElementById('output-7-2').innerHTML = `
    <p><strong>Nombre completo inicial (getter):</strong> ${person.fullName}</p>
    <p><strong>Nombre completo después de setter:</strong> ${person.fullName}</p>
    <p><strong>Edad válida asignada a userWithAge:</strong> ${userWithAge.age}</p>
    <p><strong>Edad después de intentar inválida:</strong> ${userWithAge.age}</p>
```

JavaScript Avanzado - Gestión de Propiedades

7.1 Banderas y Descriptores de Propiedad

Aquí exploraremos cómo las propiedades de un objeto poseen atributos especiales (banderas) que rigen su comportamiento, y cómo podemos inspeccionar y modificar estos descriptores.

Descriptor 'user.name' por defecto:

```
{
  "value": "DANIEL",
  "writable": true,
  "enumerable": true,
  "configurable": true
}
```