EVIDENCIAS 8

```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
```

```
tipos,herencia 8 > main.js > ...
// main.js - Prácticas del Curso JavaScript.info

// Función auxiliar para registrar en consola y DOM
function logOutput(sectionId, message, isDOMResult = false) {
    const output = document.getElementById(`output-${sectionId}`);
    const result = document.getElementById(`result-${sectionId}`);
    if (output && !output.innerHTML.includes('<strong>Salida de Consola')) output.innerHTML = `<strong>S
    if (output) {
        output.innerHTML += `<p>${message.replace(/\n/g, '<br>')}</p>`;
        output.scrollTop = output.scrollHeight;
    }
    if (isDOMResult && result && !result.innerHTML.includes('<strong>Resultados en el DOM')) result.inne
    if (isDOMResult && result) {
        result.innerHTML += `<p>${message}</p>`;
        result.scrollTop = result.scrollHeight;
    }
    console.log(`[${sectionId}] ${message}`);
}

// ================================================================================
// TEMA: 8.1 Herencia Prototípica
// Demuestra cómo los objetos heredan propiedades y métodos de sus prototipos.

logOutput('8-1', "--- Iniciando 8.1 Herencia Prototípica ---");

// Objeto 'animal' como prototipo.
let animal = {
    eats: true,
    walk() { logOutput('8-1', "El animal camina.", true); }
};
logOutput('8-1', "Objeto 'animal' definido.");

// 'rabbit' hereda de 'animal'.
let rabbit = Object.create(animal);
rabbit.jumps = true;
logOutput('8-1', "Objeto 'rabbit' creado heredando de 'animal'.");
```

```
// Sobreescribir método heredado.
rabbit.walk = function() { logOutput('8-1', "El conejo salta y camina enérgicamente!", true); };
logOutput('8-1', "Método 'walk' sobreescrito en 'rabbit'.");
rabbit.walk();

// Cadena de prototipos y `hasOwnProperty`.
logOutput('8-1', `Prototipo de rabbit === animal: ${rabbit.__proto__ === animal}`);
logOutput('8-1', `Propiedades de 'rabbit' (propia/heredada):`);
for (let prop in rabbit) {
    logOutput('8-1', `  ${prop}: ${rabbit[prop]} (Propia: ${rabbit.hasOwnProperty(prop)})`, true);
}
logOutput('8-1', "--- Fin 8.1 Herencia Prototípica ---");

// ================================================================================
// TEMA: 8.2 Prototipo F (funciones constructoras)
// Explora el uso de `F.prototype` para herencia con funciones constructoras.

logOutput('8-2', "--- Iniciando 8.2 Prototipo F ---");

// Función constructora 'Person'.
function Person(name) { this.name = name; }

// Métodos y propiedades en `Person.prototype`.
Person.prototype.sayHi = function() { logOutput('8-2', `Hola, soy ${this.name} desde el prototipo.`, tru
Person.prototype.species = "Homo Sapiens";
logOutput('8-2', "Función constructora 'Person' y su prototipo definidos.");

// Creación de instancias.
let john = new Person("John");
let anna = new Person("Anna");
logOutput('8-2', "Instancias 'john' y 'anna' creadas.");

// Acceso a métodos y propiedades.
```

```
logOutput('8-2', `Especie de Anna (sigue heredando): ${anna.species}`, true);
logOutput('8-2', "--- Fin 8.2 Prototipo F ---");

// ================================================================================
// TEMA: 8.3 Prototipos Nativos
// Describe cómo los objetos nativos usan prototipos y cómo extenderlos (con cautela).

logOutput('8-3', "--- Iniciando 8.3 Prototipos Nativos ---");

// Array.prototype
let myArray = [1, 2, 3];
logOutput('8-3', `myArray hereda de Array.prototype: ${myArray.__proto__ === Array.prototype}`);

// Extender Array.prototype (¡con precaución!).
if (!Array.prototype.lastUpper) {
    Array.prototype.lastUpper = function() {
        if (this.length === 0) return "Array vacío";
        const last = this[this.length - 1];
        return typeof last === 'string' ? last.toUpperCase() : last;
    };
}
logOutput('8-3', "Método 'lastUpper' añadido a Array.prototype.");
let fruits = ["apple", "banana", "kiwi"];
logOutput('8-3', `Última fruta en mayúsculas: ${fruits.lastUpper()}`, true);

// String.prototype
let myString = "hello javascript";
logOutput('8-3', `myString hereda de String.prototype: ${myString.__proto__ === String.prototype}`);

// Extender String.prototype.
if (!String.prototype.reverseString) {
    String.prototype.reverseString = function() { return this.split('').reverse().join(''); };
}

ring' añadido a String.prototype.");
"${myString.reverseString()}"`, true);
```

```
Lín. 47, col. 1  Espacios: 4  UTF-8
```

JavaScript Avanzado - Prototipos y Herencia

Guía Importante:
Para una visualización completa y un análisis detallado de los ejemplos de código, por favor, abre la consola de tu navegador. Utiliza `F12` (en Windows/Linux) o `Cmd + Opt + I` (en macOS). Es recomendable recargar la página después de abrir la consola para asegurar la captura de todos los logs.

8. Herencia Prototípica

8.1 Herencia Prototípica

Este concepto fundamental en JavaScript explica cómo los objetos pueden heredar propiedades y métodos de otros objetos, formando una "cadena de prototipos". Cuando se accede a una propiedad que no está directamente en un objeto, JavaScript busca sucesivamente en su prototipo, y luego en el prototipo de este, hasta llegar a `null`.

```
¿Rabbit duerme? (propiedad heredada de animal): true
¿Rabbit salta? (propiedad propia de rabbit): true
El animal camina.
```