EVIDENCIAS 4

```html
<div class="section-container" id="tema-4-2">
    <h2>4.2 Referencias y Copia de Objetos</h2>
    <div class="content">
        <p>Este apartado ilustra el comportamiento de las referencias en la copia de objetos, contra
        <h3>Resultados:</h3>
        <div class="result-box">
            <p>Observa la consola (F12) para comprender las diferencias entre referencias y copias d
            <p id="output-4-2"></p>
        </div>
    </div>
</div>

<div class="section-container" id="tema-4-3">
    <h2>4.3 Recolección de Basura (Garbage Collection)</h2>
    <div class="content">
        <p>Explicación conceptual de cómo JavaScript gestiona la memoria y libera recursos de objeto
        <h3>Consideraciones:</h3>
        <div class="result-box">
            <p>Este tema es conceptual; la comprensión radica en cómo el motor de JavaScript optimiz
        </div>
    </div>
</div>

<div class="section-container" id="tema-4-4">
    <h2>4.4 Métodos de Objeto y "this"</h2>
    <div class="content">
        <p>Análisis de la creación de métodos dentro de objetos y el uso fundamental de la palabra c
        <h3>Resultados:</h3>
        <div class="result-box">
            <p>Verifica la consola (F12) para observar la ejecución de métodos y el contexto de `thi
            <p id="output-4-4"></p>
        </div>
    </div>
```

```
// TEMA: 4.1 Objetos
// Creación, acceso, modificación, adición y eliminación de propiedades de objetos.
// Iteración sobre propiedades.
console.log("--- TEMA 4.1: Objetos ---");
let usuario = { nombre: "linda", edad: 15, "es Admin": true };
console.log("Nombre:", usuario.nombre, "Edad:", usuario["edad"], "¿Es Admin?", usuario["es Admin"]);
usuario.edad = 31; // Modificar
usuario.email = "juan@example.com"; // Añadir
delete usuario.edad; // Eliminar
console.log("¿Tiene nombre?", "nombre" in usuario, "Usuario:", usuario);
for (let key in usuario) console.log(`${key}: ${usuario[key]}`);
document.getElementById('output-4-1').innerText = `Nombre: ${usuario.nombre}, Email: ${usuario.email}`
// Fin TEMA 4.1

// TEMA: 4.2 Referencias de objetos y copia
// Los objetos se asignan por referencia. Métodos para copia superficial (Object.assign, spread).
console.log("\n--- TEMA 4.2: Referencias de objetos y copia ---");
let user = { name: "sofia", age: 25 };
let admin = user; // Referencia
admin.age = 26;
console.log("Edad de user (modificado por admin):", user.age);
console.log("¿Son user y admin el mismo objeto?", user === admin); // true

let clon = Object.assign({}, user); // Copia superficial con Object.assign
clon.name = "Clonado";
console.log("Nombre user original:", user.name, "Nombre clon:", clon.name);

let clon3 = { ...user }; // Copia superficial con spread
clon3.age = 50;
console.log("Edad user original:", user.age, "Edad clon3:", clon3.age);
document.getElementById('output-4-2').innerText = `user.age: ${user.age}, clon.name: ${clon.name}`;
// Fin TEMA 4.2

// TEMA: 4.3 Recolección de basura
// JavaScript gestiona automáticamente la memoria, liberando objetos inalcanzables.
console.log("\n--- TEMA 4.3: Recolección de basura ---");
let obj1 = { data: "datos" };
```

```
// TEMA: 4.5 Constructor, operador "new"
// Funciones constructoras para crear múltiples objetos similares usando `new`.
console.log("\n--- TEMA 4.5: Constructor, operador 'new' ---");
function Usuario(nombre, edad) {
    this.nombre = nombre;
    this.edad = edad;
    this.saludar = function() { console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`); };
}
let usuario1 = new Usuario("linda", 15);
let usuario2 = new Usuario("sofia", 35);
console.log("Usuario 1:", usuario1);
usuario1.saludar();
usuario2.saludar();
document.getElementById('output-4-5').innerText = "Resultados en la consola. Objetos creados con constru
// Fin TEMA 4.5

// TEMA: 4.6 Encadenamiento opcional "?."
// Acceso seguro a propiedades anidadas; devuelve `undefined` si un intermedio es `null`/`undefined`.
console.log("\n--- TEMA 4.6: Encadenamiento opcional '?.' ---");
let userProfile = { name: "galeano", address: { street: "Calle Falsa 123" }, contact: null };
console.log("Ciudad:", userProfile.address.street);
console.log("Teléfono (con ?. ):", userProfile.contact?.phone); // undefined
console.log("Código postal (con ?. ):", userProfile.address?.zipCode); // undefined

let adminUser = { name: "Boss", greet() { console.log("Hola, soy el jefe."); } };
let guestUser = {};
adminUser.greet?.(); // Llama si existe
guestUser.greet?.(); // No hace nada
document.getElementById('output-4-6').innerText = "Resultados en la consola. Ver el uso de '?.' para evi
// Fin TEMA 4.6

// TEMA: 4.7 Tipo de símbolo
// `Symbol` es un tipo primitivo único para claves de propiedades que evitan colisiones de nombres.
console.log("\n--- TEMA 4.7: Tipo de símbolo ---");
let id = Symbol("id");
let id2 = Symbol("id");
```

```
let globalId = Symbol.for("globalId"); // Symbol global
let globalIdAgain = Symbol.for("globalId");
console.log("¿Symbols globales iguales?", globalId === globalIdAgain); // true
console.log("Descripción Symbol global:", Symbol.keyFor(globalId));
document.getElementById('output-4-7').innerText = "Resultados en la consola. Ver el uso y unic
// Fin TEMA 4.7

// TEMA: 4.8 Conversión de objeto a primitivo
// Cómo los objetos se convierten a valores primitivos (string, number) en ciertos contextos.
// Se usan `Symbol.toPrimitive`, `toString()`, `valueOf()`.
console.log("\n--- TEMA 4.8: Conversión de objeto a primitivo ---");
let producto = {
    nombre: "Manzana",
    precio: 1.5,
    toString() { return this.nombre; },
    valueOf() { return this.precio; },
    [Symbol.toPrimitive](hint) {
        if (hint == "string") return `Producto: ${this.nombre} ($${this.precio})`;
        if (hint == "number") return this.precio;
        return this.nombre + " - " + this.precio;
    }
};
console.log("Producto como string:", `${producto}`);
console.log("Producto como número:", +producto);
console.log("Producto + 2:", producto + 2);

let libro = { titulo: "El Gran Gato", paginas: 300 };
console.log("Libro como string:", String(libro)); // Usa toString()
console.log("Libro + 10:", libro + 10); // Llama a toString()
document.getElementById('output-4-8').innerText = "Resultados en la consola. Ver cómo los obje
// Fin TEMA 4.8
```

JavaScript Avanzado - Gestión de Objetos

4.1 Fundamentos de Objetos

Exploración de la creación y acceso a propiedades de objetos en JavaScript.

Resultados:

Consulta la consola del navegador (F12) para los resultados detallados de la manipulación de objetos.

Resultados en la consola. Nombre: linda, Email: juan@example.com

4.2 Referencias y Copia de Objetos

Este apartado ilustra el comportamiento de las referencias en la copia de objetos, contrastándolo con los tipos primitivos.

Resultados:

Observa la consola (F12) para comprender las diferencias entre referencias y copias de objetos.