Evidencias objetos lo básico

index.html > html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prácticas del Curso de JavaScript - Objetos</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f9f9f9;
            margin: 20px;
            color: #333;
        }
        h1 {
            color: #e83e8c; /* Rosa similar al de tu imagen */
            text-align: center;
            margin-bottom: 30px;
        }
        .section-container {
            background-color: #ffe6f2; /* Fondo rosa claro para las secciones */
            border-radius: 8px;
            padding: 20px;
            margin-bottom: 20px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
        h2 {
            color: #d63384; /* Rosa más oscuro para los títulos de sección */
            margin-top: 0;
            border-bottom: 1px solid #f0b3cc;
            padding-bottom: 10px;
            margin-bottom: 15px;
        }
        h3 {
            color: #c2185b; /* Un tono más fuerte para los subtítulos */
            margin-top: 20px;
        }
```

index.html > html

```html
<html lang="es">
<body>
    <div class="section-container" id="tema-4-3">
    </div>

    <div class="section-container" id="tema-4-4">
        <h2>4.4 Métodos de objeto, "this"</h2>
        <p>Exploración de los métodos dentro de los objetos y el uso de la palabra clave `this
        <h3>Resultado en Consola / HTML:</h3>
        <div class="result-box">
            <p>Ver la consola (F12) para los resultados de la ejecución de métodos y el valor
            <p id="output-4-4"></p>
        </div>
    </div>

    <div class="section-container" id="tema-4-5">
        <h2>4.5 Constructor, operador "new"</h2>
        <p>Aprender a usar funciones constructoras y el operador `new` para crear múltiples ob
        <h3>Resultado en Consola / HTML:</h3>
        <div class="result-box">
            <p>Ver la consola (F12) para los objetos creados con el constructor y el operador
            <p id="output-4-5"></p>
        </div>
    </div>

    <div class="section-container" id="tema-4-6">
        <h2>4.6 Encadenamiento opcional "?."</h2>
        <p>Uso del operador de encadenamiento opcional para acceder a propiedades de objetos a
        <h3>Resultado en Consola / HTML:</h3>
        <div class="result-box">
            <p>Ver la consola (F12) para demostrar cómo `?.` evita errores.</p>
            <p id="output-4-6"></p>
        </div>
    </div>

    <div class="section-container" id="tema-4-7">
        <h2>4.7 Tipo de símbolo</h2>
```

```
console.log("Edad del user original:", user.age); // Sigue siendo 26
console.log("Edad del clon3:", clon3.age); // Es 50

document.getElementById('output-4-2').innerText = "Resultados en la consola. user.age: " + use
// Fin TEMA 4.2

// TEMA: 4.3 Recolección de basura — 24/10/2025
// Este tema es principalmente conceptual. JavaScript tiene un recolector de basura
// que automáticamente libera la memoria ocupada por objetos que ya no son accesibles
// o referenciados por ninguna parte del programa. Esto evita fugas de memoria y
// simplifica la gestión de la memoria para el desarrollador.
// No hay un código directo para "mostrar" la recolección de basura, ya que es un proceso inte
// Aprendimos que JS gestiona la memoria automáticamente cuando un objeto se vuelve "inalcanza
console.log("\n--- TEMA 4.3: Recolección de basura ---");
let obj1 = { data: "datos importantes" };
let obj2 = obj1; // obj2 referencia a obj1

obj1 = null; // obj1 ya no referencia al objeto, pero obj2 sí
console.log("obj2 todavía tiene los datos:", obj2.data);

obj2 = null; // Ahora el objeto original ya no tiene referencias, será elegible para recolecci
console.log("El objeto original ahora es inalcanzable y será recolectado.");
document.getElementById('tema-4-3').getElementsByTagName('h3')[0].insertAdjacentHTML('afterend
// Fin TEMA 4.3

// TEMA: 4.4 Métodos de objeto, "this" — 24/10/2025
// Este bloque introduce los métodos (funciones dentro de objetos) y cómo usar la palabra clav
// `this` se refiere al objeto actual donde se ejecuta el método.
// Aprendimos cómo definir funciones como propiedades de objetos y el contexto de `this`.
console.log("\n--- TEMA 4.4: Métodos de objeto, 'this' ---");
let persona = {
    nombre: "santos",
    saludar: function() {
        // `this` se refiere al objeto 'persona'
        console.log("Hola, mi nombre es " + this.nombre);
    },
```

main.js > userProfile > name

```
document.getElementById('output-4-4').innerText = "Resultados en la consola. Ver los saludos y
// Fin TEMA 4.4

// TEMA: 4.5 Constructor, operador "new" — 24/10/2025
// Este bloque explica cómo crear funciones constructoras para generar múltiples objetos con l
// El operador `new` se usa para invocar la función constructora, la cual:
// 1. Crea un nuevo objeto vacío.
// 2. Asigna `this` a ese nuevo objeto.
// 3. Ejecuta el cuerpo de la función.
// 4. Retorna `this`.
// Aprendimos a estandarizar la creación de objetos similares.
console.log("\n--- TEMA 4.5: Constructor, operador 'new' ---");

// Función constructora
function Usuario(nombre, edad) {
    this.nombre = nombre;
    this.edad = edad;
    this.saludar = function() {
        console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);
    };
}

// Crear instancias de Usuario usando 'new'
let usuario1 = new Usuario("linda", 15);
let usuario2 = new Usuario("sofia", 35);

console.log("Usuario 1:", usuario1);
usuario1.saludar();

console.log("Usuario 2:", usuario2);
usuario2.saludar();

// Es importante usar 'new'. Sin 'new', 'this' apuntaría al objeto global (window)
// let usuarioSinNew = Usuario("Error", 10); // Esto intentaría modificar window.nombre y wind
// console.log("Usuario sin new:", window.nombre); // Solo visible si no estás en modo estrict
```

main.js > userProfile > name

```
// TEMA: 4.6 Encadenamiento opcional "?." — 24/10/2025
// El operador de encadenamiento opcional (`?.`) permite acceder a propiedades
// anidadas de objetos sin tener que comprobar explícitamente si cada nivel
// de la cadena de propiedades existe. Si un intermedio es `null` o `undefined`,
// la expresión se detiene y devuelve `undefined` en lugar de lanzar un error.
// Aprendimos a escribir código más seguro y conciso al acceder a propiedades profundas.
console.log("\n--- TEMA 4.6: Encadenamiento opcional '?.' ---");

let userProfile = {
    name: "galeano",
    address: {
        street: "Calle Falsa 123",
        city: "Springfield"
    },
    contact: null // Esta propiedad es nula
};

// Acceso normal a propiedades existentes
console.log("Ciudad:", userProfile.address.city); // Springfield

// Intento de acceder a una propiedad de una propiedad nula (provoca error sin '?.')
// console.log("Teléfono:", userProfile.contact.phone); // TypeError: Cannot read properties o

// Con encadenamiento opcional, evita el error
console.log("Teléfono (con ?. ):", userProfile.contact?.phone); // undefined
console.log("Código postal (con ?. ):", userProfile.address?.zipCode); // undefined (la propie

// Se puede usar también con llamadas a métodos
let adminUser = {
    name: "Boss",
    greet() {
        console.log("Hola, soy el jefe.");
    }
};
```

Prácticas del Curso de JavaScript - Objetos: Lo Básico

4.1 Objetos

Aquí se exploran los fundamentos de los objetos en JavaScript, cómo se crean y cómo acceder a sus propiedades.

Resultado en Consola / HTML:

Ver la consola (F12) para los resultados de la creación y acceso a propiedades de objetos.

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

4.2 Referencias de objetos y copia

Este apartado muestra cómo los objetos se copian por referencia y las diferencias con los tipos primitivos.

Resultado en Consola / HTML:

Ver la consola (F12) para entender cómo funcionan las referencias y copias de objetos.

Resultados en la consola. user.age: 26, clon2.name: Clonado

4.3 Recolección de basura