EVIDENCIAS 14

```
<!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 */
```

```
// main.js - Código del curso JavaScript.info
// Nombre: [Tu Nombre Completo]
// Curso: [Tu Grado y Grupo]
// Fecha: [Fecha Actual]

console.log("main.js cargado correctamente.");
document.addEventListener('DOMContentLoaded', () => {

    // Helper para actualizar el DOM y la consola
    const updateOutput = (id, message, isError = false) => {
        const element = document.getElementById(id);
        if (element) {
            element.innerHTML += `<p style="${isError ? 'color: red;' : ''}">${message}</p
        }
        if (isError) console.error(message);
        else console.log(message);
    };

    // ================================================================
    // TEMA: 14.1 Proxy y Reflejo (Reflect)
    // `Proxy` intercepta operaciones en objetos; `Reflect` provee operaciones por defecto
    // Aprendí: Controlar y observar el comportamiento de los objetos.
    // ================================================================
    console.log("--- 14.1 Proxy y Reflect ---");
    const output14_1 = document.getElementById('output-14-1');
    if (output14_1) output14_1.innerHTML = "Revisa la consola para Proxy y Reflect.";

    // Ejemplo de Proxy
    let user = {
        name: "Alice"
    };
    let handler = {
        get(target, prop, receiver) {
            console.log(`GET ${prop}`); // Intercepta la lectura de propiedades
```

```
// ================================================================
// TEMA: 14.3 Currificación (Curry)
// Transformar `f(a,b,c)` en `f(a)(b)(c)`. Crea funciones con argumentos parciales.
// Aprendí: Técnica funcional para reutilización y composición de funciones.
// ================================================================
console.log("\n--- 14.3 Currificación ---");

function curry(func) {
    return function curried(...args) {
        if (args.length >= func.length) {
            return func.apply(this, args);
        } else {
            return function(...args2) {
                return curried.apply(this, args.concat(args2));
            };
        }
    };
}

function sum(a, b, c) {
    return a + b + c;
}
let curriedSum = curry(sum);

updateOutput('output-14-3', `sum(1, 2, 3): ${curriedSum(1, 2, 3)}`); // 6
updateOutput('output-14-3', `curriedSum(1)(2)(3): ${curriedSum(1)(2)(3)}`); // 6
updateOutput('output-14-3', `curriedSum(1, 2)(3): ${curriedSum(1, 2)(3)}`); // 6
updateOutput('output-14-3', `curriedSum(1)(2, 3): ${curriedSum(1)(2, 3)}`); // 6

// ================================================================
// TEMA: 14.4 Tipo de Referencia (Reference Type)
```

```
// ================================================================
// TEMA: 14.5 `BigInt`: Números Enteros Grandes
// Tipo de dato para enteros de precisión arbitraria, más allá de `Number`.
// Sufijo `n` (ej: `123n`). Operaciones requieren que ambos operandos sean `BigInt`.
// Aprendí: Manejar números enteros muy grandes sin pérdida de precisión.
// ================================================================
console.log("\n--- 14.5 BigInt ---");

const largeNumber = 9007199254740991n; // Máximo seguro para Number + 1
const anotherBigInt = 10n;

updateOutput('output-14-5', `Número normal (MAX_SAFE_INTEGER): ${Number.MAX_SAFE_INTEGER}`);
updateOutput('output-14-5', `BigInt (MAX_SAFE_INTEGER + 1): ${largeNumber}`);
updateOutput('output-14-5', `Suma: ${largeNumber} + ${anotherBigInt} = ${largeNumber + another
updateOutput('output-14-5', `Multiplicación: ${largeNumber} * ${anotherBigInt} = ${largeNumbe

try {
    largeNumber + 1; // Error: No se pueden mezclar BigInt y Number directamente en operacion
} catch (e) {
    updateOutput('output-14-5', `Error esperado: ${e.message} (BigInt + Number)`, true);
}

// ================================================================
// TEMA: 14.6 Unicode y Componentes Internos de Cadenas
// JS usa UTF-16. Caracteres fuera del BMP (`\u{1F600}`) usan pares sustitutos.
// `length` y `for...of` se comportan diferente.
// Aprendí: Manejo preciso de caracteres Unicode en JS.
// ================================================================
console.log("\n--- 14.6 Unicode y Cadenas ---");
```

Calidad del Curso - Temas Varios de JavaScript

<F12> para ver la mayoría de los resultados en la consola del navegador.

14.1 Proxy y Reflejo (Reflect)

Aquí se demostrará el uso de los objetos `Proxy` para interceptar y personalizar operaciones fundamentales en objetos, así como el objeto `Reflect` para operaciones predeterminadas.

```
Revisa la consola para Proxy y Reflect.
```

14.2 `eval()`: Ejecutando Cadenas de Código

Se mostrará cómo la función global `eval()` puede ejecutar código JavaScript proporcionado como una cadena. Se destacarán sus usos y las consideraciones de seguridad.

```
            Resultados de la ejecución de `eval()` en la consola.

Código ejecutado: 'let x = 10; x * 2;' => Resultado: 20
```