EVIDENCIAS 12

```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
```

```javascript
// 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', () => {

    // ================================================================
    // TEMA: 12.1 Generadores
    // Funciones que pausan y reanudan su ejecución con `yield`, creando iteradores.
    // Aprendí: Crear iteradores personalizados de forma sencilla.
    // ================================================================
    console.log("--- 12.1 Generadores ---");
    const output12_1 = document.getElementById('output-12-1');

    function* idGenerator() {
        let id = 1;
        while (true) {
            yield id++;
        }
    }

    const generator = idGenerator();
    let idContent = "<h3>Ejemplo de Generadores (idGenerator)</h3>";
    for (let i = 0; i < 3; i++) {
        idContent += `<p>ID: ${generator.next().value}</p>`;
    }
    if (output12_1) output12_1.innerHTML = idContent;
    console.log("12.1 Generadores: IDs generados por consola:", generator.next().value, generator.nex

    // Generador Fibonacci
    function* fibonacciGenerator() {
```

```javascript
    // ================================================================
    // TEMA: 12.2 Async iteration and generators
    // Iteración y generadores asíncronos para secuencias de valores resueltas de forma asíncrona.
    // Aprendí: Uso de `for await...of` y `async function*`.
    // ================================================================
    console.log("\n--- 12.2 Async iteration and generators ---");
    const output12_2 = document.getElementById('output-12-2');

    async function* asyncNumberGenerator() {
        let num = 1;
        while (num <= 5) {
            await new Promise(resolve => setTimeout(resolve, 300)); // Simula retardo
            yield num++;
        }
    }

    if (output12_2) {
        output12_2.innerHTML = "<h3>Ejemplo de Generadores Asíncronos</h3><p>Generando números asíncr

        (async () => {
            let numContent = "<p>Números generados:</p><ul>";
            for await (let number of asyncNumberGenerator()) {
                numContent += `<li>${number}</li>`;
                output12_2.innerHTML = "<h3>Ejemplo de Generadores Asíncronos</h3>" + numContent; //
                console.log("12.2 Async iteration: Número:", number);
            }
            numContent += "</ul><p>Generación asíncrona completada.</p>";
            output12_2.innerHTML = "<h3>Ejemplo de Generadores Asíncronos</h3>" + numContent;
        })();
    } else {
        console.warn("12.2 Async iteration: Elemento 'output-12-2' no encontrado.");
    }
});
```

Iteración Avanzada en JavaScript

12. Generadores e Iteración Avanzada

12.1 Introducción a los Generadores

Este bloque demuestra la implementación y el uso de funciones generadoras en JavaScript, una herramienta poderosa para controlar la ejecución de funciones y manejar secuencias de datos.

```
Cargando resultados del generador... (Revisa la consola para detalles)
```

12.2 Iteración y Generadores Asíncronos

Este bloque explora las capacidades de la iteración asíncrona y los generadores asíncronos, fundamentales para trabajar con flujos de datos asíncronos de manera eficiente y legible.

```
Cargando resultados de iteración asíncrona... (Revisa la consola para detalles)
```