Evidencias manejo de errores 9

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Clases en JavaScript</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #fce4ec; /* Rosa claro */
            margin: 0;
            padding: 20px;
        }
        h1 {
            color: #d81b60; /* Rosa oscuro */
            text-align: center;
            margin-bottom: 30px;
        }
        .section-container {
            background-color: #fff;
            border: 1px solid #f8bbd0; /* Rosa pastel */
            border-radius: 8px;
            padding: 20px;
            margin-bottom: 20px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);
        }
        h2 {
            color: #c2185b; /* Rosa medio */
            border-bottom: 1px solid #f48fb1; /* Rosa claro */
            padding-bottom: 10px;
            margin-bottom: 15px;
        }
        pre {
            background-color: #f8f8f8;
            border: 1px solid #eee;
            padding: 10px;
            border-radius: 4px;
```

```
            padding: 10px;
            margin-top: 15px;
            border-radius: 4px;
            white-space: pre-wrap; /* Para mantener los saltos de línea */
        }
    </style>
</head>
<body>
    <h1>Manejo de Errores en JavaScript</h1>

    <div class="section-container" id="tema-10-1">
        <h2>10.1 Manejo de errores, "try...catch"</h2>
        <p>Aquí aprenderemos a usar la estructura `try...catch` para manejar errores y evitar que la eje
        <div class="output" id="output-10-1">
            <!-- La salida de JS se mostrará aquí -->
        </div>
    </div>

    <div class="section-container" id="tema-10-2">
        <h2>10.2 Errores personalizados, error de extensión</h2>
        <p>En esta sección, veremos cómo crear nuestros propios tipos de errores personalizados para mar
        <div class="output" id="output-10-2">
            <!-- La salida de JS se mostrará aquí -->
        </div>
    </div>

    <script src="main.js"></script>
</body>
</html>
```

```
log10_1 += "\nEl programa continúa después del bloque try...catch...finally.\n";
output10_1.textContent = log10_1;

// Ejemplo práctico: Parsar JSON
log10_1 += "\n--- Ejemplo práctico: Parsar JSON ---\n";
let json = '{ "nombre": "Alice", "edad": 30 }';
try {
    let user = JSON.parse(json); // Parsea el JSON correctamente
    log10_1 += `Usuario parseado: ${user.nombre}, ${user.edad} años.\n`;

    let badJson = '{ "edad": 25 }'; // JSON incompleto para simular un error al acceder a 'nombre'
    let badUser = JSON.parse(badJson);
    // Intentamos acceder a una propiedad que podría no existir, aunque el JSON.parse funcione
    log10_1 += `Usuario con JSON incompleto (intentando acceder a nombre): ${badUser.nombre}\n`;
    // Nota: Acceder a badUser.nombre no lanza un error aquí, simplemente es undefined.
    // Un error se lanzaría si badUser fuera null/undefined y se intentara acceder a una propiedad.
    // Para demostrar un error de parseo:
    // let invalidJson = '{ "nombre": "Bob" }'; // JSON mal formado
    // JSON.parse(invalidJson); // Esto sí lanzaría un SyntaxError
} catch (error) {
    log10_1 += `¡Error al parsear JSON o acceder a propiedades! ${error.name}: ${error.message}\n`;
} finally {
    output10_1.textContent = log10_1;
}


// TEMA: 10.2 Errores personalizados, error de extensión - 20/05/2024
// Este bloque explica cómo crear y usar errores personalizados.
// Es útil cuando queremos diferenciar errores específicos de nuestra lógica de negocio.

const output10_2 = document.getElementById('output-10-2');
let log10_2 = "--- Tema 10.2: Errores personalizados ---\n\n";

// Paso 1: Definir una clase para nuestro error personalizado
```

Lín. 150, col. 54 Espacios: 4 UTF-8 CRLF {} JavaSc

```
log10_2 += "Clases ValidationError y NetworkError definidas.\n\n";

// Paso 2: Usar nuestro error personalizado en un bloque try...catch
function verificarEdad(edad) {
    if (isNaN(edad) || edad < 0) {
        throw new ValidationError("La edad debe ser un número positivo.");
    }
    if (edad < 18) {
        throw new ValidationError("Debe ser mayor de 18 años.");
    }
    return true;
}

try {
    log10_2 += "Intentando verificar edad (20 años)...\n";
    verificarEdad(20);
    log10_2 += "Edad verificada correctamente: 20 años.\n\n";

    log10_2 += "Intentando verificar edad (15 años)...\n";
    verificarEdad(15); // Esto lanzará un ValidationError
    log10_2 += "Esta línea no se ejecutará si la edad es menor a 18.\n";

} catch (error) {
    if (error instanceof ValidationError) {
        log10_2 += `¡Error de Validación capturado! Nombre: ${error.name}, Mensaje: ${error.me
        // Podemos añadir lógica específica para este tipo de error
    } else {
        log10_2 += `¡Otro tipo de error capturado! Nombre: ${error.name}, Mensaje: ${error.mes
    }
    log10_2 += "La ejecución continúa después de capturar el error de validación.\n\n";
}
```

Manejo de Errores en JavaScript

10.1 Manejo de errores, "try...catch"

Aquí aprenderemos a usar la estructura `try...catch` para manejar errores y evitar que la ejecución del script se detenga inesperadamente.

```
--- Tema 10.1: try...catch ---

Iniciando bloque try...
Código dentro de try ejecutado sin error aparente.
¡Error capturado en el bloque catch!
Tipo de error: Error
Mensaje de error: ¡Algo salió mal intencionalmente!
La ejecución del programa continúa después del catch.

Bloque finally siempre ejecutado.

El programa continúa después del bloque try...catch...finally.

--- Ejemplo práctico: Parsar JSON ---
Usuario parseado: Alice, 30 años.
Usuario con JSON incompleto (intentando acceder a nombre): undefined
```

10.2 Errores personalizados, error de extensión

En esta sección, veremos cómo crear nuestros propios tipos de errores personalizados para manejar situaciones específicas en nuestra aplicación.

```
--- Tema 10.2: Errores personalizados ---

Clases ValidationError y NetworkError definidas.
```