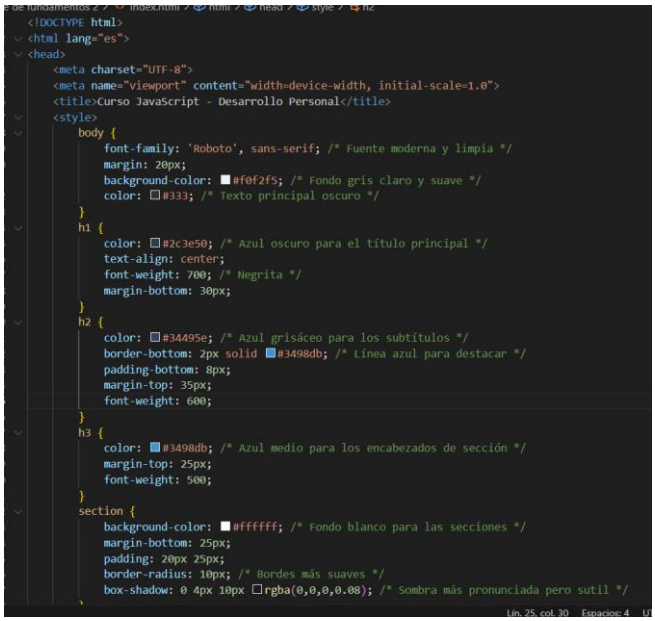
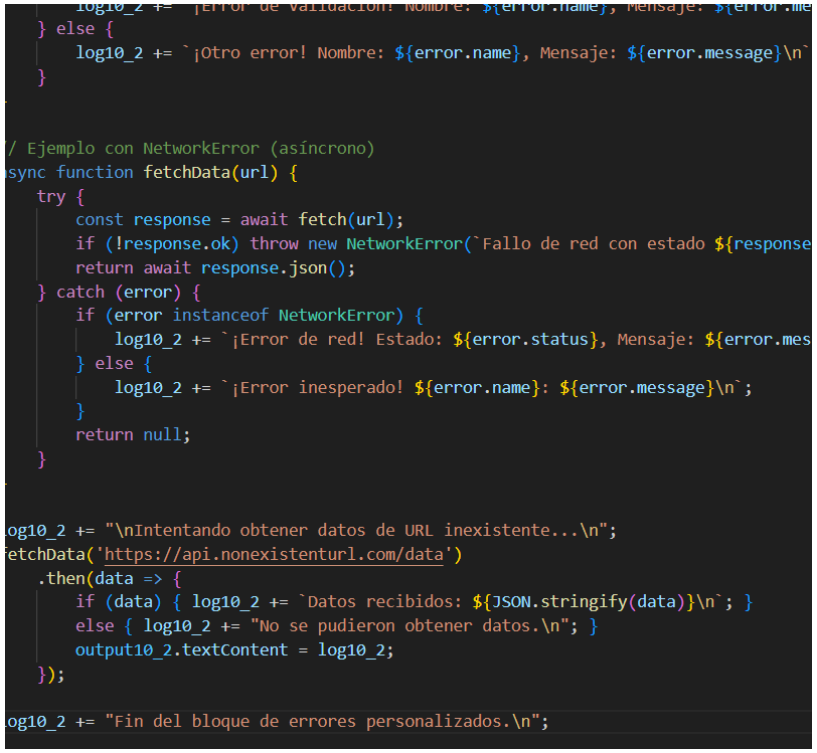
EVIDENCIAS 10



```
<!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 */
```

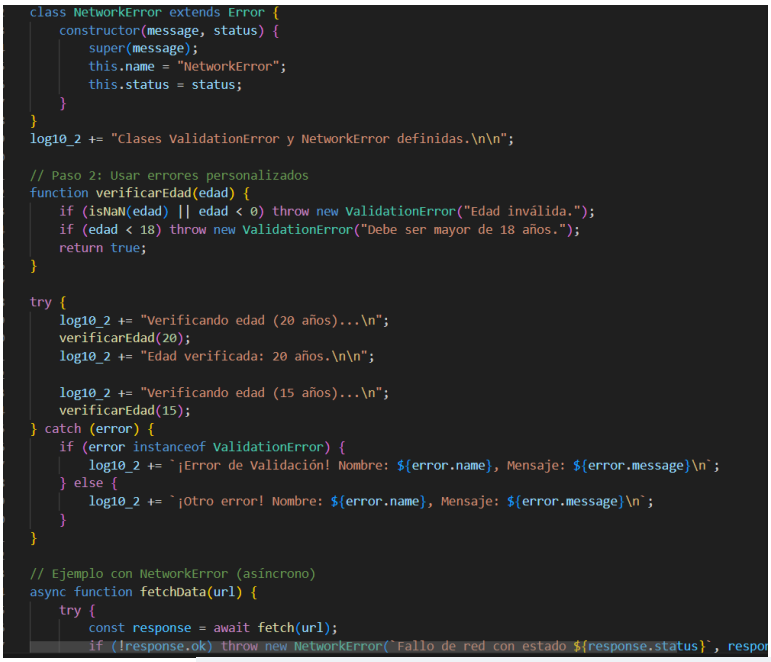


```
    } else {
        log10_2 += `¡Otro error! Nombre: ${error.name}, Mensaje: ${error.message}\n`
    }


// Ejemplo con NetworkError (asíncrono)
async function fetchData(url) {
    try {
        const response = await fetch(url);
        if (!response.ok) throw new NetworkError(`Fallo de red con estado ${response
        return await response.json();
    } catch (error) {
        if (error instanceof NetworkError) {
            log10_2 += `¡Error de red! Estado: ${error.status}, Mensaje: ${error.mes
        } else {
            log10_2 += `¡Error inesperado! ${error.name}: ${error.message}\n`;
        }
        return null;
    }


log10_2 += "\nIntentando obtener datos de URL inexistente...\n";
fetchData('https://api.nonexistenturl.com/data')
    .then(data => {
        if (data) { log10_2 += `Datos recibidos: ${JSON.stringify(data)}\n`; }
        else { log10_2 += "No se pudieron obtener datos.\n"; }
        output10_2.textContent = log10_2;
    });

log10_2 += "Fin del bloque de errores personalizados.\n";
```



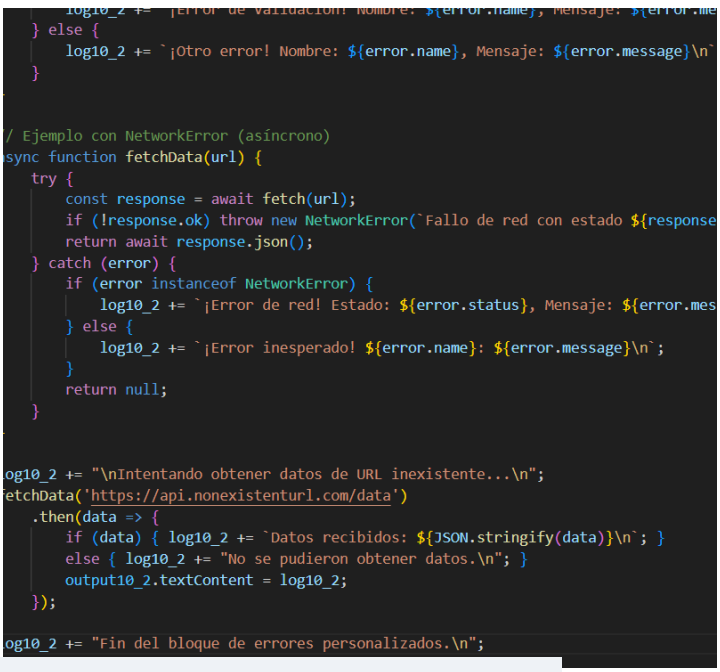
```
class NetworkError extends Error {
    constructor(message, status) {
        super(message);
        this.name = "NetworkError";
        this.status = status;
    }
}
log10_2 += "Clases ValidationError y NetworkError definidas.\n\n";

// Paso 2: Usar errores personalizados
function verificarEdad(edad) {
    if (isNaN(edad) || edad < 0) throw new ValidationError("Edad inválida.");
    if (edad < 18) throw new ValidationError("Debe ser mayor de 18 años.");
    return true;
}

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

    log10_2 += "Verificando edad (15 años)...\n";
    verificarEdad(15);
} catch (error) {
    if (error instanceof ValidationError) {
        log10_2 += `¡Error de Validación! Nombre: ${error.name}, Mensaje: ${error.message}\n`;
    } else {
        log10_2 += `¡Otro error! Nombre: ${error.name}, Mensaje: ${error.message}\n`;
    }
}

// Ejemplo con NetworkError (asíncrono)
async function fetchData(url) {
    try {
        const response = await fetch(url);
        if (!response.ok) throw new NetworkError(`Fallo de red con estado ${response.status}`, respor
```



```
    } else {
        log10_2 += `¡Otro error! Nombre: ${error.name}, Mensaje: ${error.message}\n`
    }


// Ejemplo con NetworkError (asíncrono)
async function fetchData(url) {
    try {
        const response = await fetch(url);
        if (!response.ok) throw new NetworkError(`Fallo de red con estado ${response
        return await response.json();
    } catch (error) {
        if (error instanceof NetworkError) {
            log10_2 += `¡Error de red! Estado: ${error.status}, Mensaje: ${error.mes
        } else {
            log10_2 += `¡Error inesperado! ${error.name}: ${error.message}\n`;
        }
        return null;
    }


log10_2 += "\nIntentando obtener datos de URL inexistente...\n";
fetchData('https://api.nonexistenturl.com/data')
    .then(data => {
        if (data) { log10_2 += `Datos recibidos: ${JSON.stringify(data)}\n`; }
        else { log10_2 += "No se pudieron obtener datos.\n"; }
        output10_2.textContent = log10_2;
    });

log10_2 += "Fin del bloque de errores personalizados.\n";
```

JavaScript - Gestión de Errores

10.1 Gestión de Errores con "try...catch"

En esta sección, se explorará el uso fundamental de la estructura `try...catch` para interceptar y manejar excepciones, asegurando que la ejecución del script continúe sin interrupciones inesperadas.

```
--- 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
```