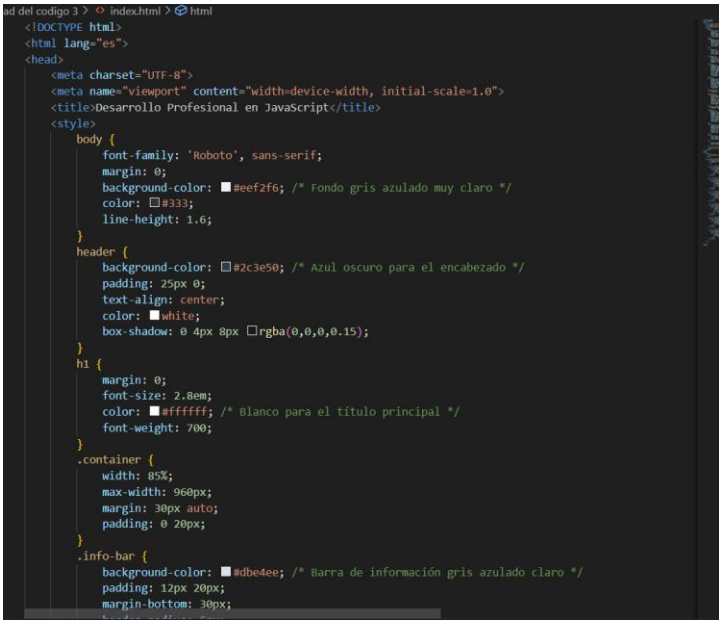
EVIDENCIAS 3

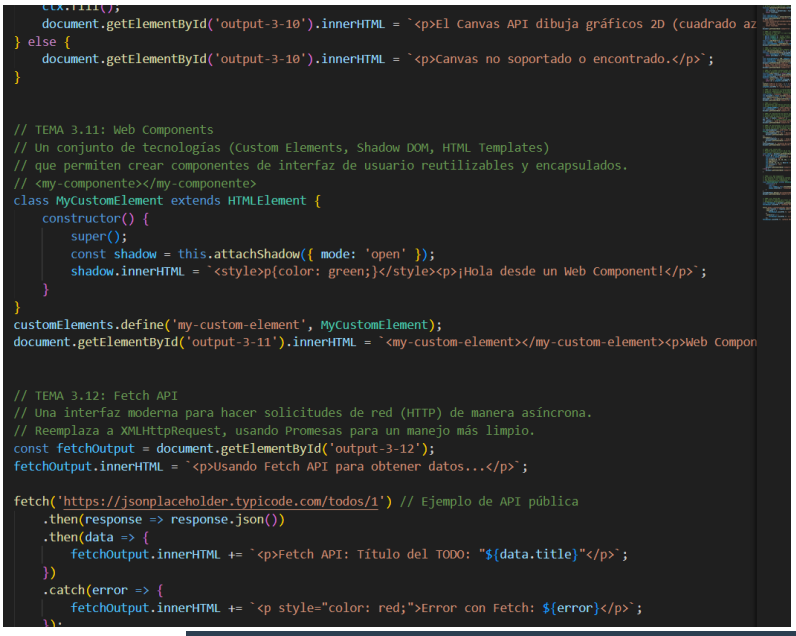


```
ad del codigo 3 > index.html > html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Desarrollo Profesional en JavaScript</title>
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            margin: 0;
            background-color: #eef2f6; /* Fondo gris azulado muy claro */
            color: #333;
            line-height: 1.6;
        }
        header {
            background-color: #2c3e50; /* Azul oscuro para el encabezado */
            padding: 25px 0;
            text-align: center;
            color: white;
            box-shadow: 0 4px 8px rgba(0,0,0,0.15);
        }
        h1 {
            margin: 0;
            font-size: 2.8em;
            color: #ffffff; /* Blanco para el título principal */
            font-weight: 700;
        }
        .container {
            width: 85%;
            max-width: 960px;
            margin: 30px auto;
            padding: 0 20px;
        }
        .info-bar {
            background-color: #dbe4ee; /* Barra de información gris azulado claro */
            padding: 12px 20px;
            margin-bottom: 30px;
```

```
    return precioBase * (1 + tasaImpuesto);
}
const precioProducto = 100, impuesto = 21;
const totalNinja = calcularTotalNinja(precioProducto, impuesto / 100);
const totalLegible = calcularPrecioFinal(precioProducto, impuesto);
document.getElementById('code-3-4').textContent = `
// Ninja: const calcularTotalNinja = (p, c) => p * (1 + c);
// Legible: function calcularPrecioFinal(...) {...}`;
document.getElementById('output-3-4').innerHTML = `<p>Precio ninja: ${totalNinja.toFixe

// TEMA 3.5: Pruebas automatizadas con Mocha (simulación)
// Pruebas aseguran que el código funcione y evitan errores.
function esPar(numero) { return numero % 2 === 0; }
const output35 = document.getElementById('output-3-5');
output35.innerHTML = '<h3>Simulación de pruebas: `esPar()`</h3>';
function runTest(desc, func) {
    try { func(); output35.innerHTML += `<p style="color: green;">✓ ${desc}</p>`; }
    catch (e) { output35.innerHTML += `<p style="color: red;">X ${desc} - ${e.message}
}
runTest('esPar(2) es true', () => { if (!esPar(2)) throw new Error('Falló para 2'); });
runTest('esPar(3) es false', () => { if (esPar(3)) throw new Error('Falló para 3'); });
output35.innerHTML += '<p>Las pruebas reales usarían frameworks como Mocha/Chai.</p>';

// TEMA 3.6: Polyfills y transpiladores
// Aseguran compatibilidad de JS moderno con navegadores antiguos.
// Polyfills: implementan funcionalidades. Transpiladores (Babel): convierten código (
const frutas = ['manzana', 'banana'];
let mensaje36 = frutas.includes('banana') ? '<p>Array incluye "banana" (moderno).</p>'
const miFuncionFlecha = () => "Función flecha.";
mensaje36 += `<p>${miFuncionFlecha()}</p><p>Transpiladores adaptan código moderno a ES
document.getElementById('output-3-6').innerHTML = mensaje36;
```



```
    ctx.fill();
    document.getElementById('output-3-10').innerHTML = `<p>El Canvas API dibuja gráficos 2D (cuadrado az
} else {
    document.getElementById('output-3-10').innerHTML = `<p>Canvas no soportado o encontrado.</p>`;
}


// TEMA 3.11: Web Components
// Un conjunto de tecnologías (Custom Elements, Shadow DOM, HTML Templates)
// que permiten crear componentes de interfaz de usuario reutilizables y encapsulados.
// <my-componente></my-componente>
class MyCustomElement extends HTMLElement {
    constructor() {
        super();
        const shadow = this.attachShadow({ mode: 'open' });
        shadow.innerHTML = `<style>p{color: green;}</style><p>¡Hola desde un Web Component!</p>`;
    }
}
customElements.define('my-custom-element', MyCustomElement);
document.getElementById('output-3-11').innerHTML = `<my-custom-element></my-custom-element><p>Web Compon


// TEMA 3.12: Fetch API
// Una interfaz moderna para hacer solicitudes de red (HTTP) de manera asíncrona.
// Reemplaza a XMLHttpRequest, usando Promesas para un manejo más limpio.
const fetchOutput = document.getElementById('output-3-12');
fetchOutput.innerHTML = `<p>Usando Fetch API para obtener datos...</p>`;

fetch('https://jsonplaceholder.typicode.com/todos/1') // Ejemplo de API pública
    .then(response => response.json())
    .then(data => {
        fetchOutput.innerHTML += `<p>Fetch API: Título del TODO: "${data.title}"</p>`;
    })
    .catch(error => {
        fetchOutput.innerHTML += `<p style="color: red;">Error con Fetch: ${error}</p>`;
    });
```