Evidencias de clases 9

index.html > html > head

```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, similar a tu imagen */
            margin: 0;
            padding: 20px;
        }
        h1 {
            color: #d81b60; /* Rosa más oscuro para el título */
            text-align: center;
            margin-bottom: 30px;
        }
        section {
            background-color: #ffe0f0; /* Rosa aún más claro para las secciones */
            border-radius: 8px;
            padding: 20px;
            margin-bottom: 20px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
        h2 {
            color: #ad1457; /* Otro tono de rosa para subtítulos */
            border-bottom: 2px solid #f8bbd0;
            padding-bottom: 10px;
            margin-top: 0;
        }
        pre {
            background-color: #f0f0f0;
            padding: 10px;
            border-radius: 4px;
            overflow-x: auto;
        }
```

```html
        <div class="output" id="output-9-1"></div>
    </section>

    <section id="clases-9-2">
        <h2>9.2 Herencia de clase</h2>
        <p>Este ejemplo demuestra cómo una clase puede heredar propiedades y métodos de otra clase.</p>
        <div class="output" id="output-9-2"></div>
    </section>

    <section id="clases-9-3">
        <h2>9.3 Propiedades y métodos estáticos</h2>
        <p>Aquí se muestra el uso de propiedades y métodos que pertenecen a la clase misma, no a sus ins
        <div class="output" id="output-9-3"></div>
    </section>

    <section id="clases-9-4">
        <h2>9.4 Propiedades y métodos privados y protegidos</h2>
        <p>Este bloque ilustra cómo se pueden usar las convenciones de `_` (protegidos) y `#` (privados)
        <div class="output" id="output-9-4"></div>
    </section>

    <section id="clases-9-5">
        <h2>9.5 Ampliación de clases integradas</h2>
        <p>Ejemplo de cómo extender clases de JavaScript ya existentes, como `Array` o `Error`.</p>
        <div class="output" id="output-9-5"></div>
    </section>

    <section id="clases-9-6">
        <h2>9.6 Comprobación de clase: "instanceof"</h2>
        <p>Uso del operador `instanceof` para verificar si un objeto es una instancia de una clase espec
        <div class="output" id="output-9-6"></div>
    </section>
```

```javascript
// TEMA: 9.3 Propiedades y métodos estáticos – 23/10/2023
// Este bloque ilustra el uso de métodos y propiedades estáticas que pertenecen a la clase, no a las
// Aprendí que los miembros estáticos se acceden directamente desde la clase (e.g., Matematica.PI) y
class Matematica {
    static PI = 3.14159; // Propiedad estática

    static sumar(a, b) { // Método estático
        return a + b;
    }

    static restar(a, b) { // Otro método estático
        return a - b;
    }
}

// Acceso a propiedades y métodos estáticos
const suma = Matematica.sumar(5, 3);
const resta = Matematica.restar(10, 4);
const piValor = Matematica.PI;

const output9_3 = document.getElementById('output-9-3');
output9_3.innerHTML += `<p>La suma de 5 y 3 es: ${suma}</p>`;
output9_3.innerHTML += `<p>La resta de 10 y 4 es: ${resta}</p>`;
output9_3.innerHTML += `<p>El valor de PI es: ${piValor}</p>`;
console.log("9.3 Propiedades y métodos estáticos: Suma =", suma, "Resta =", resta, "PI =", piValor);
```

```javascript
const miCuenta = new CuentaBancaria("Juan Perez", 1000);
miCuenta.depositar(200);
miCuenta.retirar(500);

const output9_4 = document.getElementById('output-9-4');
output9_4.innerHTML += `<p>Titular: ${miCuenta._titular}</p>`;
output9_4.innerHTML += `<p>Saldo actual: ${miCuenta.getSaldo()}</p>`;
output9_4.innerHTML += `<p>${miCuenta.getNumeroCuentaVisible()}</p>`;
output9_4.innerHTML += `<p class="console-output">Ver la consola para los detalles de depósitos
console.log("9.4 Propiedades y métodos privados y protegidos: Titular =", miCuenta._titular, "S
// Intentar acceder a miCuenta.#saldo o miCuenta.#generarNumeroCuenta() directamente fuera de l


// TEMA: 9.5 Ampliación de clases integradas – 23/10/2023
// Este bloque muestra cómo extender una clase nativa de JavaScript, en este caso, Array.
// Aprendí que puedo añadir funcionalidades personalizadas a tipos de datos existentes, lo cual
class MiArrayExtendido extends Array {
    sumarTodos() {
        return this.reduce((acc, current) => acc + current, 0);
    }

    obtenerPares() {
        return this.filter(num => num % 2 === 0);
    }
}

const numeros = new MiArrayExtendido(1, 2, 3, 4, 5, 6);
const sumaTotal = numeros.sumarTodos();
const pares = numeros.obtenerPares();

const output9_5 = document.getElementById('output-9-5');
output9_5.innerHTML += `<p>Array original: [${numeros.join(', ')}]</p>`;
```

```javascript
// TEMA: 9.6 Comprobación de clase: "instanceof" – 23/10/2023
// Este bloque utiliza el operador 'instanceof' para verificar si un objeto es una instancia de un
// Aprendí que 'instanceof' es crucial para la verificación de tipos en herencia y polimorfismo, y
class Animal { }
class Perro extends Animal { }
class Gato extends Animal { }

const unPerro = new Perro();
const unGato = new Gato();
const unaPersona = new Persona("Laura", 40); // Reutilizamos la clase Persona de 9.1

const output9_6 = document.getElementById('output-9-6');
output9_6.innerHTML += `<p>unPerro instanceof Perro: ${unPerro instanceof Perro}</p>`; // true
output9_6.innerHTML += `<p>unPerro instanceof Animal: ${unPerro instanceof Animal}</p>`; // true
output9_6.innerHTML += `<p>unPerro instanceof Object: ${unPerro instanceof Object}</p>`; // true
output9_6.innerHTML += `<p>unGato instanceof Perro: ${unGato instanceof Perro}</p>`;   // false
output9_6.innerHTML += `<p>unaPersona instanceof Persona: ${unaPersona instanceof Persona}</p>`; /
output9_6.innerHTML += `<p>unaPersona instanceof Estudiante: ${unaPersona instanceof Estudiante}</
console.log("9.6 Comprobación de clase: instanceof. Ver resultados en HTML.");


// TEMA: 9.7 Mixins – 23/10/2023
// Este bloque demuestra la implementación de mixins en JavaScript para añadir funcionalidades reu
// Aprendí que los mixins son una forma de lograr reutilización de código similar a la herencia mú
// Se usan para "mezclar" comportamientos en una clase.
let SaludarMixin = {
    decirHola() {
        console.log(`Hola, soy ${this.nombre}.`);
    },
    decirAdios() {
        console.log(`Adiós de parte de ${this.nombre}.`);
    }
```

Clases en JavaScript

9.1 Sintaxis básica de clase

Aquí verás la implementación de una clase simple y cómo se crean objetos a partir de ella.

Hola, mi nombre es Alice y tengo 30 años.
Hola, mi nombre es Bob y tengo 25 años.

9.2 Herencia de clase

Este ejemplo demuestra cómo una clase puede heredar propiedades y métodos de otra clase.

Hola, soy Carlos, tengo 20 años y estudio Ingeniería de Sistemas.
Carlos está estudiando Ingeniería de Sistemas.

9.3 Propiedades y métodos estáticos

Aquí se muestra el uso de propiedades y métodos que pertenecen a la clase misma, no a sus instancias.

La suma de 5 y 3 es: 8
La resta de 10 y 4 es: 6