EVIDENCIAS TIPOS DE DATOS

index.html > html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Práctica JavaScript - Tipos de Datos</title>
    <style>
        /* Estilos generales del cuerpo */
        body {
            font-family: Arial, sans-serif;
            background-color: #fce4ec; /* Rosa muy claro */
            margin: 0;
            padding: 20px;
            color: #333;
        }

        /* Estilos del encabezado */
        header {
            background-color: #f8bbd0; /* Rosa un poco más oscuro */
            padding: 20px;
            text-align: center;
            border-radius: 8px;
            margin-bottom: 30px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }

        header h1 {
            color: #ad1457; /* Rosa oscuro */
            margin-bottom: 10px;
        }

        header h2 {
            color: #d81b60; /* Rosa medio */
        }

        /* Estilos para cada sección de tema */
```

index.html > html

```html
<html lang="es">
<body>
    <header>
        <h1>Prácticas de JavaScript</h1>
        <h2>Tipos de Datos</h2>
    </header>

    <main>
        <section id="tema-5-1" class="topic-section">
            <h3>5.1 Métodos de primitivos</h3>
            <p>Aquí se mostrarán los resultados y ejemplos de los métodos de tipos de datos pr
            <div id="output-5-1" class="code-output"></div>
        </section>

        <section id="tema-5-2" class="topic-section">
            <h3>5.2 Números</h3>
            <p>Ejemplos y resultados relacionados con el manejo de números en JavaScript.</p>
            <div id="output-5-2" class="code-output"></div>
        </section>

        <section id="tema-5-3" class="topic-section">
            <h3>5.3 Cuerdas (Strings)</h3>
            <p>Demostración de propiedades y métodos de las cadenas de texto.</p>
            <div id="output-5-3" class="code-output"></div>
        </section>

        <section id="tema-5-4" class="topic-section">
            <h3>5.4 Matrices (Arrays)</h3>
            <p>Ejemplos de creación, acceso y manipulación de arrays.</p>
            <div id="output-5-4" class="code-output"></div>
        </section>

        <section id="tema-5-5" class="topic-section">
            <h3>5.5 Métodos de matriz</h3>
            <p>Uso de los métodos más comunes de los arrays como push, pop, map, filter, etc.<
            <div id="output-5-5" class="code-output"></div>
        </section>
```

```javascript
// TEMA: 5.3 Cuerdas (Strings) — 24/05/2024
// Este bloque cubre el tipo de dato 'String', cómo crearlos, concatenarlos,
// acceder a caracteres individuales y utilizar algunos de sus métodos más com
function runStringsExamples() {
    console.log("\n--- 5.3 Cuerdas (Strings) ---");
    let singleQuote = 'Esto es un string con comillas simples.';
    let doubleQuote = "Esto es un string con comillas dobles.";
    let backticks = `Esto es un string con backticks.`; // Para plantillas lit

    console.log(singleQuote);
    console.log(doubleQuote);
    console.log(backticks);

    // Concatenación
    let greeting = "Hola";
    let name = "Ana";
    let message = greeting + ", " + name + "!";
    console.log("Concatenación:", message); // Aprendí: El operador '+' une st

    // Plantillas literales (template literals)
    let age = 30;
    let templateMessage = `Hola, mi nombre es ${name} y tengo ${age} años.`;
    console.log("Plantilla literal:", templateMessage); // Aprendí: Las planti

    // Acceso a caracteres
    let myString = "JavaScript";
    console.log("Primer carácter (myString[0]):", myString[0]); // J
    console.log("Longitud (myString.length):", myString.length); // 10
    // Aprendí: Los strings son inmutables; no se puede cambiar un carácter di

    // Métodos de string
    console.log("Parte del string (myString.substring(0, 4)):", myString.substr
    console.log("Reemplazar 'Script' por 'Code' (myString.replace('Script', 'C
    console.log("Convertir a minúsculas (myString.toLowerCase()):", myString.t
```

```javascript
// ==========================================================================
// TEMA: 5.1 Métodos de primitivos — Resumen
// Aprendizaje clave: JavaScript permite que los tipos de datos primitivos (ej. strings, numbers
// usen métodos como si fueran objetos, convirtiéndolos temporalmente.
// Ejemplos: `str.toUpperCase()`, `num.toFixed(2)`.
// ==========================================================================
console.log("5.1 Métodos de primitivos: Primitivos usan métodos de objeto temporalmente.");

// ==========================================================================
// TEMA: 5.2 Números — Resumen
// Aprendizaje clave: Manejo de enteros y flotantes. Conversiones con `Number()`.
// Propiedades `NaN` (Not a Number) e `Infinity`. Funciones globales `isNaN()` y `isFinite()`.
// Uso del objeto `Math` para operaciones como `floor()`, `ceil()`, `round()`, `random()`.
// Importancia de la precisión de punto flotante (`0.1 + 0.2` no siempre es `0.3`).
// ==========================================================================
console.log("5.2 Números: Conversiones, NaN/Infinity, Math object y precisión.");

// ==========================================================================
// TEMA: 5.3 Cuerdas (Strings) — Resumen
// Aprendizaje clave: Las cadenas son inmutables (los métodos devuelven nuevas cadenas).
// Propiedad `length`. Acceso a caracteres (`[0]`, `charAt()`).
// Búsqueda: `indexOf()`, `includes()`, `startsWith()`, `endsWith()`.
// Extracción: `slice()`, `substring()` (preferidos).
// Modificación: `replace()`. Transformación: `toUpperCase()`, `toLowerCase()`.
// Limpieza: `trim()`.
// ==========================================================================
console.log("5.3 Cuerdas: Inmutabilidad, longitud, acceso, búsqueda, extracción y transformación

// ==========================================================================
// TEMA: 5.4 Matrices (Arrays) — Resumen
// Aprendizaje clave: Arrays como colecciones ordenadas, mutables y con índices base 0.
// Acceso y modificación por índice.
// Métodos para agregar/eliminar en extremos: `push()`, `unshift()`, `pop()`, `shift()`.
```

```javascript
sole.log("5.6 Iterables: for...of para strings/arrays. Objetos no iterables directamente. Array.from(

// ==========================================================================
TEMA: 5.7 Mapa y conjunto (Map y Set) — Resumen
Aprendizaje clave:
Map`: Colección de clave-valor. Las claves pueden ser de CUALQUIER tipo (a diferencia de objetos).
 Métodos: `set()`, `get()`, `has()`, `delete()`, `size`. Es iterable.
Set`: Colección de valores ÚNICOS. Ignora duplicados.
 Métodos: `add()`, `has()`, `delete()`, `size`. Es iterable.
// ==========================================================================
sole.log("5.7 Mapa y conjunto: Map (clave-valor con claves de cualquier tipo), Set (valores únicos).

// ==========================================================================
TEMA: 5.8 Mapa débil y conjunto débil (WeakMap y WeakSet) — Resumen
Aprendizaje clave: Versiones "débiles" de Map y Set.
Claves (`WeakMap`) o valores (`WeakSet`) deben ser OBJETOS.
No impiden la recolección de basura de esos objetos si no hay otras referencias fuertes.
No son iterables y no tienen `size`. Usos para metadatos sin evitar limpieza de memoria.
// ==========================================================================
sole.log("5.8 Mapa débil y conjunto débil: WeakMap/WeakSet (claves/valores objetos), no evitan recole

// ==========================================================================
TEMA: 5.9 Objeto.claves, valores, entradas (Object.keys, values, entries) — Resumen
Aprendizaje clave: Métodos estáticos para extraer partes de objetos literales.
Object.keys()`: Devuelve un array con las claves (propiedades).
Object.values()`: Devuelve un array con los valores.
Object.entries()`: Devuelve un array de `[clave, valor]` pares.
Esenciales para iterar y manipular propiedades de objetos.
```

EVIDENCIAS TIPOS DE DATOS

```
console.log("5.9 Objeto.claves, valores, entradas: Object.keys(), Object.values(), Object.entries() para interactua

// ========================================================================
// TEMA: 5.10 Asignación de desestructuración (Destructuring assignment) — Resumen
// Aprendizaje clave: Sintaxis para "desempaquetar" valores de arrays o propiedades de objetos
// directamente en variables separadas, de forma más concisa.
// Desestructuración de arrays: `[a, b] = [1, 2]`.
// Desestructuración de objetos: `{ nombre, edad } = usuario`.
// Soporta valores por defecto, renombrado de variables y parámetros de función.
// ========================================================================
console.log("5.10 Asignación de desestructuración: Desempaquetar valores de arrays/objetos en variables de forma co

// ========================================================================
// TEMA: 5.11 Fecha y hora (Date and time) — Resumen
// Aprendizaje clave: Uso del objeto `Date` para manejar fechas y horas.
// Crear fechas: `new Date()`, `new Date(string)`, `new Date(año, mes, ...)`.
// Métodos para obtener componentes: `getFullYear()`, `getMonth()`, `getDate()`, `getHours()`, `getMinutes()`, `get
// Métodos para establecer componentes: `setFullYear()`, `setMonth()`, etc.
// Obtener timestamp: `getTime()`.
// ========================================================================
```

```
// TEMA: 5.11 Fecha y hora (Date and time) — Resumen
// Aprendizaje clave: Uso del objeto `Date` para manejar fechas y horas.
// Crear fechas: `new Date()`, `new Date(string)`, `new Date(año, mes, ...)`.
// Métodos para obtener componentes: `getFullYear()`, `getMonth()`, `getDate()`, `getHours()`,
// Métodos para establecer componentes: `setFullYear()`, `setMonth()`, etc.
// Obtener timestamp: `getTime()`.
// ========================================================================
console.log("5.11 Fecha y hora: Objeto Date para crear, obtener y manipular fechas/horas.");

// ========================================================================
// TEMA: 5.12 Métodos JSON, toJSON — Resumen
// Aprendizaje clave: Serialización y deserialización de datos JavaScript a y desde JSON.
// `JSON.stringify()`: Convierte un objeto/array JS a una cadena JSON.
// `JSON.parse()`: Convierte una cadena JSON a un objeto/array JS.
// Método `toJSON()`: Un método opcional que un objeto puede definir para personalizar su seri
// cuando se usa `JSON.stringify()`.
// ========================================================================
console.log("5.12 Métodos JSON, toJSON: JSON.stringify() para JS a JSON, JSON.parse() para JSO
```

5.1 Métodos de primitivos

Aquí se exploran cómo los tipos de datos primitivos (números, cadenas, etc.) pueden tener métodos. Aunque no son objetos, JavaScript permite acceder a sus funcionalidades como si lo fueran, cor en objetos para usar sus métodos.

Resultado en Consola / HTML:

Ver la consola (F12) para los resultados de los métodos de primitivos.

5.2 Números