Guardar Preferencias con LocalStorage (Modo Oscuro/Claro)

Este documento explica cómo crear un interruptor de tema (modo claro/oscuro) y cómo guardar la elección del usuario en el navegador para que su preferencia se recuerde en futuras visitas.

🎯 Objetivo del Ejercicio

- Crear un botón que cambie el tema visual de la página entre claro y oscuro.
- Aprender a usar **CSS Variables** para gestionar los colores del tema de forma eficiente.
- Entender qué es `localStorage` y cómo funciona.
- Usar `localStorage.setItem()` para guardar la preferencia del usuario.
- Usar `localStorage.getItem()` para leer la preferencia guardada cuando la página se carga.
- Lograr que la página "recuerde" el modo elegido por el usuario incluso después de cerrar y volver a abrir el navegador.

📁 Archivos Necesarios

Necesitarás tres archivos en la misma carpeta:

1. `modo-oscuro.html` (La estructura de la página y el botón)
2. `modo-oscuro.css` (Los estilos para ambos temas)
3. `modo-oscuro.js` (La lógica para cambiar el tema y usar `localStorage`)

Paso 1: La Estructura HTML (`modo-oscuro.html`)

La estructura es muy simple. Solo necesitamos algo de contenido para ver el cambio de tema y un botón para activarlo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Modo Oscuro con LocalStorage</title>
    <!-- Enlazamos nuestra hoja de estilos -->
    <link rel="stylesheet" href="modo-oscuro.css">
</head>
<body>

    <header>
        <h1>Mi Página Increíble</h1>
        <button id="theme-toggle">Cambiar Tema</button>
    </header>

    <main>
```

```
        <h2>Bienvenido a mi sitio</h2>
        <p>Este es un ejemplo para demostrar cómo guardar el tema oscuro o claro usando LocalStorage. ¡Prueba a cambiar el tema, recarga la página o cierra y vuelve a abrir el navegador!</p>
        <a href="#">Este es un enlace de ejemplo</a>
    </main>

    <!-- Enlazamos nuestro archivo JavaScript al final -->
    <script src="modo-oscuro.js"></script>
</body>
</html>
```

Paso 2: Los Estilos y Variables CSS (`modo-oscuro.css`)

Aquí está la clave visual. Usaremos variables CSS. Definimos un conjunto de colores por defecto (modo claro) y luego los sobreescribimos cuando el `<body>` tenga la clase `dark-mode`.

```
/* --- 1. Definición de Variables de Color --- */
:root {
    --bg-color: #f4f4f4;
    --text-color: #333;
    --header-bg: #e2e2e2;
    --link-color: #007BFF;
}

body.dark-mode {
    --bg-color: #2c2c2c;
    --text-color: #f1f1f1;
    --header-bg: #1a1a1a;
    --link-color: #64b5f6;
}

/* --- 2. Aplicación de Estilos Usando las Variables --- */
body {
    background-color: var(--bg-color);
    color: var(--text-color);
    font-family: sans-serif;
    transition: background-color 0.3s, color 0.3s;
}

header {
    background-color: var(--header-bg);
    padding: 20px;
    text-align: center;
    border-bottom: 1px solid var(--text-color);
}
```

```css
button {
    padding: 10px 15px;
    border: 1px solid var(--text-color);
    background-color: transparent;
    color: var(--text-color);
    cursor: pointer;
}

a {
    color: var(--link-color);
}
```

Paso 3: La Lógica de JavaScript (`modo-oscuro.js`)

Este es el cerebro de la operación. El código tiene dos responsabilidades principales:

1. Reaccionar al clic del botón para cambiar el tema y guardarlo.
2. Comprobar al cargar la página si ya existe un tema guardado para aplicarlo.

```javascript
// Este archivo está vacío a propósito. ¡Tu misión es completarlo!
// Sigue los comentarios como una guía paso a paso.

// --- 1. OBTENER ELEMENTOS DEL DOM ---
// Guarda en una constante el botón con id 'theme-toggle'.
// Guarda en una constante el elemento 'body' del documento.


// --- 2. MANEJAR EL EVENTO CLICK EN EL BOTÓN ---
// Añade un 'event listener' al botón para que reaccione al evento 'click'.
// Dentro de la función del 'event listener', haz lo siguiente:
//      a. Usa classList.toggle('dark-mode') en el 'body' para añadir o quitar la
clase.
//
//      b. Ahora, guarda la preferencia del usuario en localStorage.
//         Usa una estructura 'if/else' para comprobar si el 'body' CONTIENE la
clase 'dark-mode'.
//         - Si la TIENE, guarda en localStorage la clave 'theme' con el valor
'dark'.
//         - Si NO la tiene, guarda en localStorage la clave 'theme' con el valor
'light'.


// --- 3. APLICAR EL TEMA GUARDADO AL CARGAR LA PÁGINA ---
// Crea una función llamada 'applySavedTheme'.
// Dentro de esta función, haz lo siguiente:
//      a. Obtén el valor guardado en localStorage para la clave 'theme' y guárdalo
en una constante.
```

```
//
//      b. Comprueba si el valor que obtuviste es exactamente igual a 'dark'.
//         - Si lo es, añade la clase 'dark-mode' al 'body'.

// Finalmente, llama a la función 'applySavedTheme()' al final del archivo para que
se ejecute
// tan pronto como la página se cargue.
```

🚀 Cómo Probarlo

1. Abre el archivo `3-modo-oscuro.html` en tu navegador. La página aparecerá en modo claro.
2. Haz clic en el botón "Cambiar Tema". La página debería cambiar instantáneamente a modo oscuro.
3. **Recarga la página (F5 o Ctrl+R).** La página debería permanecer en modo oscuro. ¡`localStorage` está funcionando!
4. Haz clic de nuevo en el botón para volver al modo claro.
5. **Cierra la pestaña del navegador y vuelve a abrir el archivo.** La página debería recordar tu última elección y cargarse en modo claro.

💡 Cheatsheet: Tu Guía Rápida de `localStorage` y Clases CSS

Concepto / Método	Uso y Explicación	Ejemplo de Uso
`localStorage`	Un objeto global en el navegador que permite guardar datos (como strings) que persisten incluso después de cerrar el navegador. Es un almacenamiento de **clave-valor**.	`localStorage`
`localStorage.setItem('clave', 'valor')`	Guarda un valor (siempre como string) en el `localStorage` asociado a una clave. Si la clave ya existe, su valor se sobreescribe.	`localStorage.setItem('theme', 'dark');`

Concepto / Método	Uso y Explicación	Ejemplo de Uso
`localStorage.getItem('clave')`	Recupera el valor asociado a una clave desde el `localStorage`. Si la clave no existe, devuelve `null`.	`const savedTheme = localStorage.getItem('theme');`
`localStorage.removeItem('clave')`	(Extra) Elimina un par clave-valor del `localStorage`.	`localStorage.removeItem('theme');`
`element.classList`	Una propiedad que da acceso a la lista de clases de un elemento. Tiene métodos útiles para manipularlas.	`document.body.classList`
`.classList.add('clase')`	Añade una clase a la lista de clases de un elemento.	`body.classList.add('dark-mode');`
`.classList.remove('clase')`	Elimina una clase de la lista de clases de un elemento.	`body.classList.remove('dark-mode');`
`.classList.toggle('clase')`	**Muy útil.** Si la clase no existe, la añade. Si ya existe, la elimina. Perfecto para interruptores.	`body.classList.toggle('dark-mode');`
`.classList.contains('clase')`	Devuelve `true` o `false` dependiendo de si el elemento tiene o no la clase especificada.	`if (body.classList.contains('dark-mode'))`
Variables CSS (`var()`)	Permiten definir valores reutilizables en CSS. Se definen con `--nombre-variable` y se usan con `var(--nombre-variable)`.	`background-color: var(--bg-color);`