REPORTE DE TALLER – SEMANA 4

TEMA: ESTANDARIZACIÓN Y DEPURACIÓN DE CÓDIGO

ENTREGADO POR: NATALY

DAYANA MEDINA BECERRA

INGENIERÍA DE SOFTWARE, SEMESTRE V

GitHub:

https://github.com/ndmb03/buenas_practicas-master.git

ENTREGADO A:

ROBINSON DAMIAN GOMEZ SANCHEZ

ARQUITECTURA Y DISEÑO DEL SOFTWARE

FUNDACIÓN DE ESTUDIOS SUPERIORES COMFANORTE, FESC CÚCUTA, NORTE

DE SANTANDER

04 DE NOVIEMBRE DE 2025

SECCIÓN 1: DEPURACIÓN DE ENLACES

Durante la revisión inicial del proyecto, se detectó que el archivo principal index.html no cargaba correctamente los estilos ni las funcionalidades JavaScript. Al inspeccionar el código, se encontró que los enlaces a los archivos CSS y JS estaban mal escritos. En el archivo se utilizaban los siguientes nombres:

```
1  <link rel="stylesheet" href="styles.css">
2  <script src="app-legcy.js"></script>
```

Sin embargo, los nombres reales de los archivos dentro del proyecto eran:

- style.css
- app-legacy.js

Este error provocaba que el navegador no encontrara los archivos correctos, lo que generaba que la interfaz no tuviera estilos ni respondiera a las acciones del usuario.

Para solucionar el problema, se aplicó la "Opción B: Corregir el HTML", ya que es la opción más profesional.

Se modificaron directamente las etiquetas <link> y <script> del archivo index.html para que apuntaran a los nombres correctos de los archivos. Después de la corrección, el código quedó así:

```
1  <link rel="stylesheet" href="style.css">
2  <script src="app-legacy.js"></script>
```

Una vez realizado este ajuste, la página cargó correctamente:

- Los estilos visuales se aplicaron adecuadamente desde style.css.
- Las funcionalidades de la calculadora y la lista de tareas (To-Do List) comenzaron a ejecutarse sin errores.

Este proceso de depuración permitió garantizar el correcto enlace entre los componentes del proyecto y demostró la importancia de verificar cuidadosamente las rutas y nombres de archivo dentro del código HTML.

SECCIÓN 2: GUÍAS DE ESTILO

Una Guía de Estilo es un conjunto de normas y convenciones que definen cómo debe escribirse el código fuente dentro de un proyecto de software. Su propósito principal es mantener la consistencia, legibilidad y calidad del código, especialmente cuando múltiples desarrolladores colaboran en un mismo proyecto.

La aplicación de una guía de estilo es fundamental porque:

1. **Facilita la lectura y comprensión del código:** Un formato unificado permite que cualquier desarrollador entienda rápidamente lo que hace cada parte del programa.
2. **Reduce errores y malas prácticas:** Las reglas evitan el uso de sintaxis insegura o ambigua, promoviendo el uso de estructuras más seguras y modernas.
3. **Mejora la mantenibilidad:** Cuando el código sigue estándares uniformes, es más fácil modificar, depurar o ampliar el sistema en el futuro.

En conclusión, las guías de estilo son una herramienta esencial para mantener proyectos escalables, profesionales y sostenibles a largo plazo dentro de cualquier equipo de desarrollo.

SECCIÓN 3: PROCESO DE ESTANDARIZACIÓN

Una vez que la aplicación cargó correctamente, se procedió a realizar una auditoría manual completa del archivo app-legacy.js.
El objetivo fue aplicar las reglas de la guía de estilo para estandarizar el código y mejorar su calidad, legibilidad y mantenibilidad.

Durante este proceso se aplicaron las siguientes reglas principales:

1. Regla “no-var”

Se reemplazaron todas las declaraciones de variables con la palabra clave var por let o const, dependiendo del contexto.

- **Antes:**

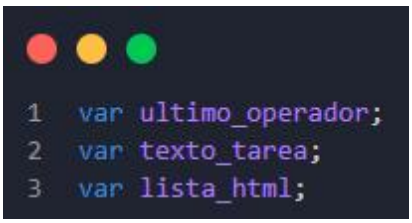
- **Después:**

Motivo:
El uso de let y const evita problemas de ámbito (scope) y hace que el código sea más predecible.
const se usa cuando el valor no cambia, mientras que let se emplea para variables cuyo valor puede modificarse.
De esta forma se mejora la seguridad y claridad del código.

2. Regla “camelCase”

Se cambiaron los nombres de variables y funciones que estaban en formato snake_case a camelCase, la convención recomendada para JavaScript moderno.

- **Antes:**



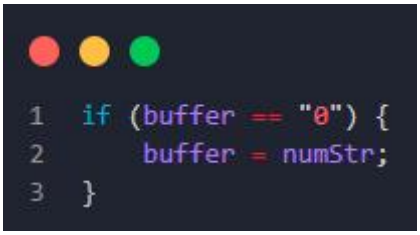
- **Después:**

Motivo:
El formato camelCase mejora la legibilidad y se alinea con las normas de estilo de ECMAScript.
Además, facilita la integración con herramientas de análisis estático (linters) y librerías modernas.

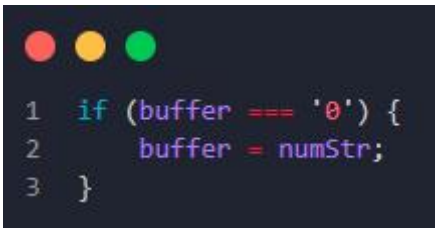
3. Regla “eqeqeq”

Se reemplazaron todos los operadores de comparación laxa (== y !=) por sus equivalentes estrictos (=== y !==).

- **Antes:**



- **Después:**

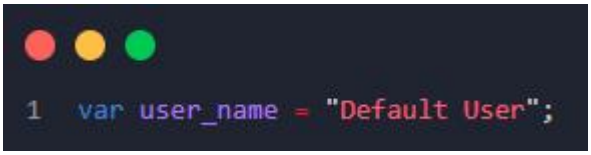


Motivo:
El operador === compara tanto el valor como el tipo de los datos, evitando errores lógicos por conversiones implícitas de tipo.
Esto hace que las condiciones sean más seguras y predecibles.

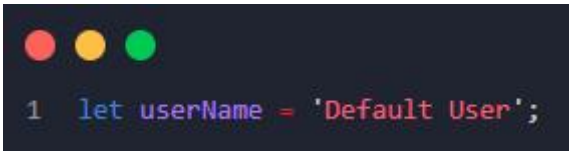
4. Regla de comillas (quotes)

Se unificó el uso de comillas en todo el archivo para mantener coherencia. Todas las comillas dobles (") se reemplazaron por comillas simples (').

- **Antes:**

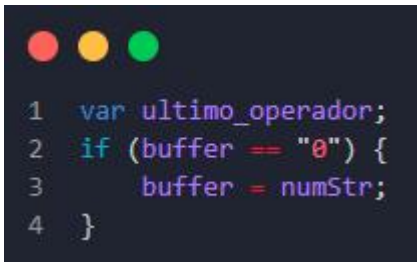


- **Después:**

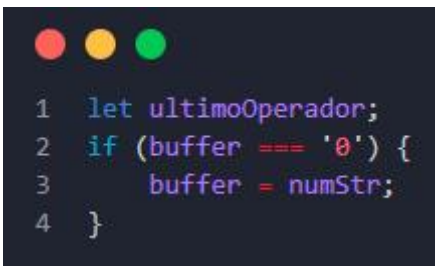


Motivo:
La uniformidad en las comillas mejora la legibilidad del código y cumple con las normas de la mayoría de guías de estilo profesionales (como la de Airbnb o Google).

Ejemplo completo de corrección aplicada Antes:



Después:



Resultado:
El código final del proyecto quedó más claro, moderno y profesional. Gracias a la estandarización, el archivo app-legacy.js ahora cumple con las buenas prácticas de desarrollo, es más fácil de mantener y su comportamiento sigue siendo funcional y confiable.

CONCLUSIONES

Este taller permitió comprender la importancia de mantener un código estandarizado y aplicar reglas de estilo dentro del desarrollo de software. La depuración de los enlaces del HTML resolvió un problema funcional inmediato, mientras que la auditoría de estilo mejoró la calidad a largo plazo del código JavaScript.

Al finalizar, se logró:

- Recuperar el funcionamiento completo de la aplicación.
- Modernizar el código heredado (legacy).
- Cumplir con los estándares de programación actuales.
- Aumentar la legibilidad, la seguridad y la mantenibilidad del proyecto