Actividad en clase: Proceso de Evaluación, Documentación y Herramientas de Calidad.

Asignatura: Arquitectura y Diseño del Software

Estudiante: Ing. Freddy Eduardo Riscanevo Mendez

Herramientas Utilizadas: Eslint – prettier

Class Activity: Evaluation Process, Documentation, and Quality Tools.

Subject: Software Architecture and Design

Student: Freddy Eduardo Riscanevo Mendez

Tools Used: Eslint – prettier

# Introducción

El presente informe expone el proceso de diagnóstico y mejora de la calidad del software aplicado al proyecto JavaScript “Calculator”, mediante el uso de herramientas profesionales de análisis y formateo de código como ESLint y Prettier. El propósito de esta práctica es identificar y corregir errores comunes de sintaxis, estilo y estructura, promoviendo un código más limpio, estandarizado y fácil de mantener. Asimismo, se busca fortalecer las competencias en el uso de entornos de desarrollo integrados (IDE) y en la aplicación de buenas prácticas de programación, fundamentales para garantizar la calidad y sostenibilidad del software tanto en contextos académicos como profesionales.

**Introduction**

This report presents the software quality diagnosis and improvement process applied to the JavaScript "Calculator" project, using professional code analysis and formatting tools such as ESLint and Prettier. The purpose of this practice is to identify and correct common syntax, style, and structure errors, promoting cleaner, more standardized, and maintainable code. It also seeks to strengthen skills in the use of integrated development environments (IDEs) and in the application of good programming practices, which are essential to ensuring software quality and sustainability in both academic and professional contexts.

**Analisis del proyecto:** La aplicación analizada corresponde a una calculadora desarrollada en JavaScript puro, concebida como un ejercicio práctico para el estudio y la mejora de la calidad del software. Este proyecto se emplea como caso de estudio en la asignatura, con el propósito de evaluar aspectos relacionados con la estructura, la legibilidad y la documentación del código fuente. A través de su análisis, se busca aplicar herramientas de diagnóstico profesional, identificar oportunidades de optimización y fortalecer las habilidades de los estudiantes en el uso de buenas prácticas de desarrollo y mantenimiento de software.

**Project Analysis:** The application analyzed is a calculator developed in pure JavaScript, conceived as a practical exercise for studying and improving software quality. This project is used as a case study in the course, with the purpose of evaluating aspects related to the structure, readability, and documentation of source code. Through its analysis, the aim is to apply professional diagnostic tools, identify optimization opportunities, and strengthen students' skills in using good software development and maintenance practices.

# Qué son estas herramientas?

* ESLint

ESLint es una herramienta fundamental dentro del ciclo profesional de desarrollo de software, orientada al análisis estático de código. Su función principal es detectar errores de sintaxis, variables no utilizadas, inconsistencias en la estructura y posibles vulnerabilidades que puedan afectar la calidad del proyecto.

Mediante la aplicación de reglas de buenas prácticas y estándares de codificación, ESLint contribuye a mejorar la mantenibilidad, legibilidad y seguridad del software. Además, su integración con editores modernos como Visual Studio Code permite automatizar la revisión del código en tiempo real, garantizando la detección temprana de errores y promoviendo una cultura de desarrollo limpio y consistente

* Prettier

Prettier es un formateador automático de código ampliamente adoptado en entornos colaborativos. Su objetivo es aplicar reglas uniformes de estilo como sangría, uso de comillas, espaciado y puntos y coma para asegurar la consistencia visual y estructural del código fuente.

A diferencia de los linters, que se enfocan en la detección de errores lógicos, Prettier se centra en la presentación del código, reduciendo discrepancias estéticas entre los miembros del equipo y mejorando la productividad en revisiones conjuntas. En conjunto con ESLint, forma una base sólida para el mantenimiento de proyectos que buscan cumplir con estándares profesionales de calidad y legibilidad.

**1. What are these tools?**

• ESLint

ESLint is a fundamental tool within the professional software development cycle, focused on static code analysis. Its main function is to detect syntax errors, unused variables, structural inconsistencies, and potential vulnerabilities that could affect project quality.

By applying good practice rules and coding standards, ESLint contributes to improving software maintainability, readability, and security. Furthermore, its integration with modern editors such as Visual Studio Code allows for real-time code review automation, ensuring early error detection and promoting a culture of clean and consistent development.

• Prettier

Prettier is an automatic code formatter widely adopted in collaborative environments. Its objective is to apply uniform style rules such as indentation, use of quotes, spacing, and semicolons to ensure visual and structural consistency in source code.

Unlike linters, which focus on detecting logical errors, Prettier focuses on code presentation, reducing aesthetic discrepancies between team members and improving productivity in collaborative reviews. In conjunction with ESLint, it forms a solid foundation for maintaining projects that strive to meet professional standards of quality and readability.

# Metodología

Para realizar el diagnóstico y mejora se siguieron los siguientes pasos:

1. Instalación y configuración de ESLint y Prettier en Visual Studio Code.
2. Ejecución de análisis y corrección automática del código fuente del proyecto Calculator.
3. Registro visual mediante capturas de pantalla que muestran el estado “Antes” y “Después” del proceso.
4. Documentación detallada de los problemas detectados y las correctivas aplicadas.

**2. Methodology**

The following steps were followed to perform the diagnosis and improvement:

1. Installation and configuration of ESLint and Prettier in Visual Studio Code.

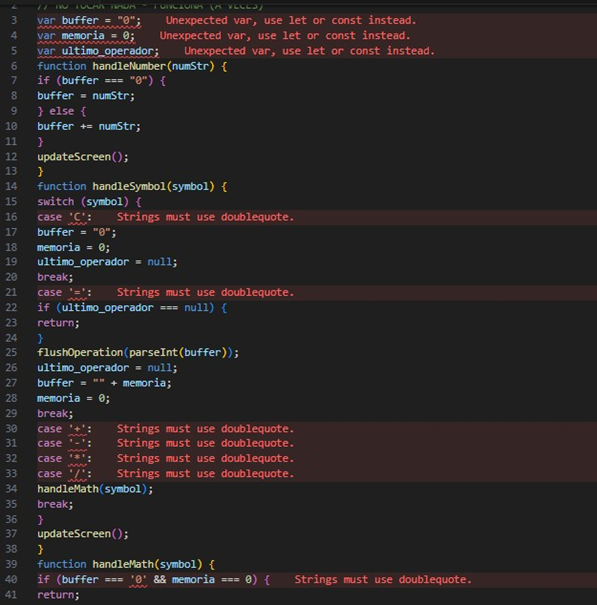
2. Automatic analysis and correction of the Calculator project source code.

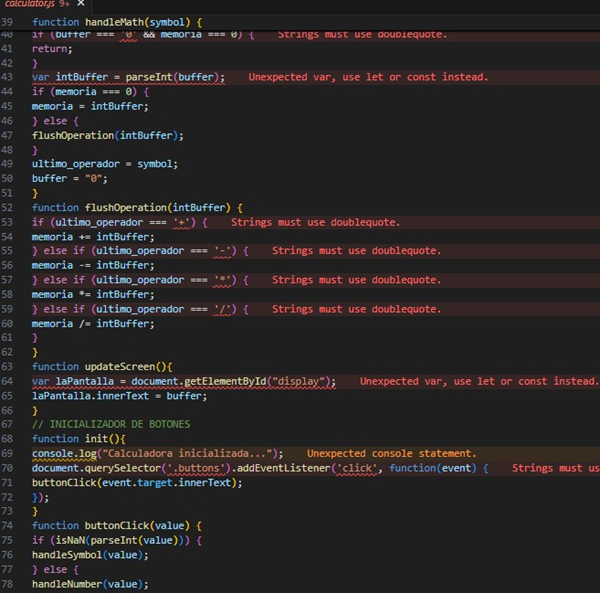
3. Visual recording using screenshots showing the "Before" and "After" status of the process.

4. Detailed documentation of the problems detected and the corrective measures applied.

**3. Evidencia Visual (ANTES) :** A primera vista, se evidencian varios errores de sintaxis y malas prácticas de codificación, que comprometen la claridad del código.

**3. Visual Evidence (BEFORE):** At first glance, several syntax errors and poor coding practices are evident, which compromise the clarity of the code.



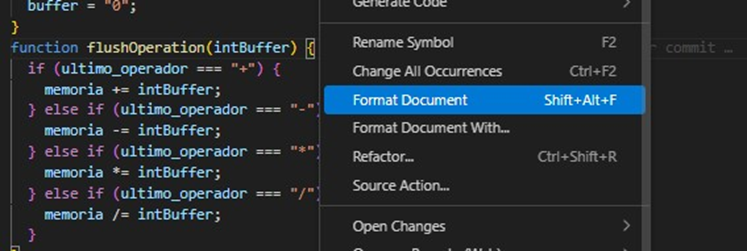


Se procedió a realizar un análisis del código utilizando la herramienta ESLint, con el propósito de identificar errores de sintaxis, advertencias y desviaciones respecto a las buenas prácticas de programación. Posteriormente, se llevó a cabo un proceso de refactorización del código, corrigiendo los aspectos señalados por la herramienta.

A code analysis was performed using the ESLint tool to identify syntax errors, warnings, and deviations from good programming practices. The code was then refactored, correcting the issues highlighted by the tool.

1. Lo primero que se realizó fue añadir formato al documento, esto mediante VSCODE dando clic derecho y seleccionando format document.

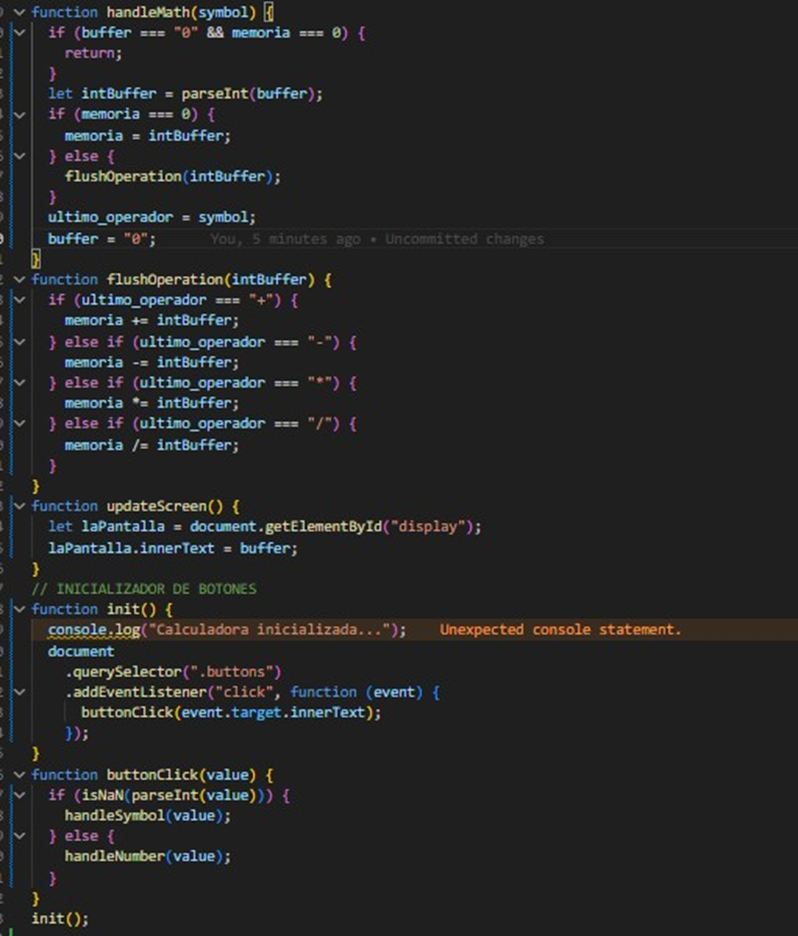
1. The first thing that was done was to add format to the document, this using VSCODE by right-clicking and selecting format document.



2. Se corrigieron los dos errores de sintaxis identificados por la herramienta ESLint. El primero correspondía al uso inadecuado de comillas simples en expresiones condicionales, lo cual generaba inconsistencias con las reglas de estilo configuradas. Se realizó la sustitución por comillas dobles en todas las secciones del código donde se presentaba este problema, asegurando así la uniformidad y cumplimiento de los estándares definidos.

2. The two syntax errors identified by the ESLint tool were corrected. The first involved the improper use of single quotes in conditional expressions, which generated inconsistencies with the configured style rules. All code sections where this problem occurred were replaced with double quotes, thus ensuring uniformity and compliance with defined standards.





3. El siguiente problema identificado por ESLint fue el uso del declarador var para la creación de variables. Dado que var posee alcance global o de función, su utilización puede provocar conflictos de nombres y comportamientos inesperados durante la ejecución del programa.

Para cumplir con las buenas prácticas modernas de JavaScript, se reemplazó por el declarador let, que ofrece un alcance de bloque más seguro y controlado, garantizando así una mejor legibilidad y mantenibilidad del código.

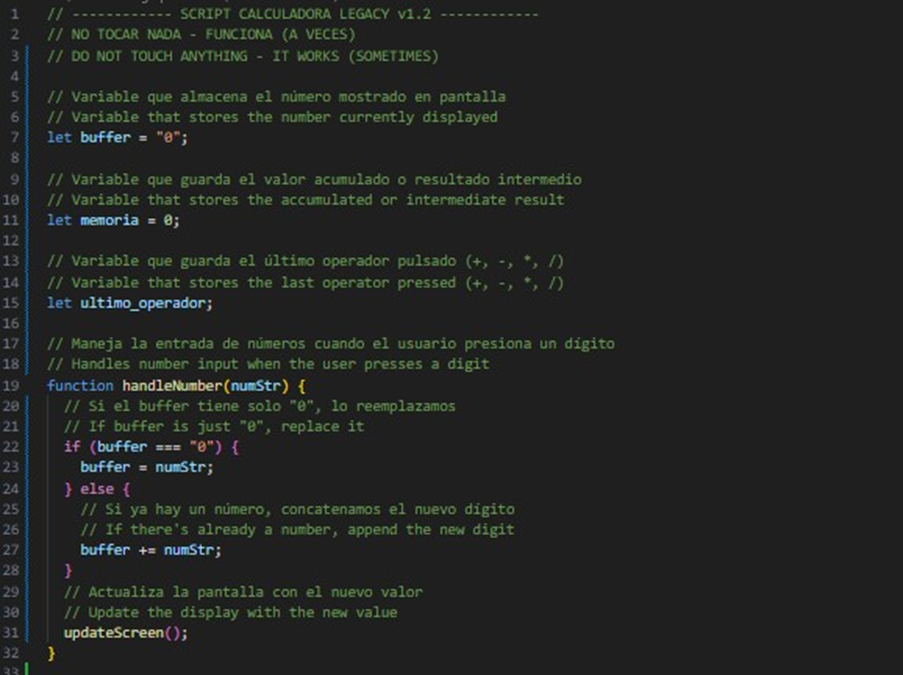
3. The next issue identified by ESLint was the use of the var declarator for variable creation. Since var has global or function scope, its use can lead to name conflicts and unexpected behavior during program execution.

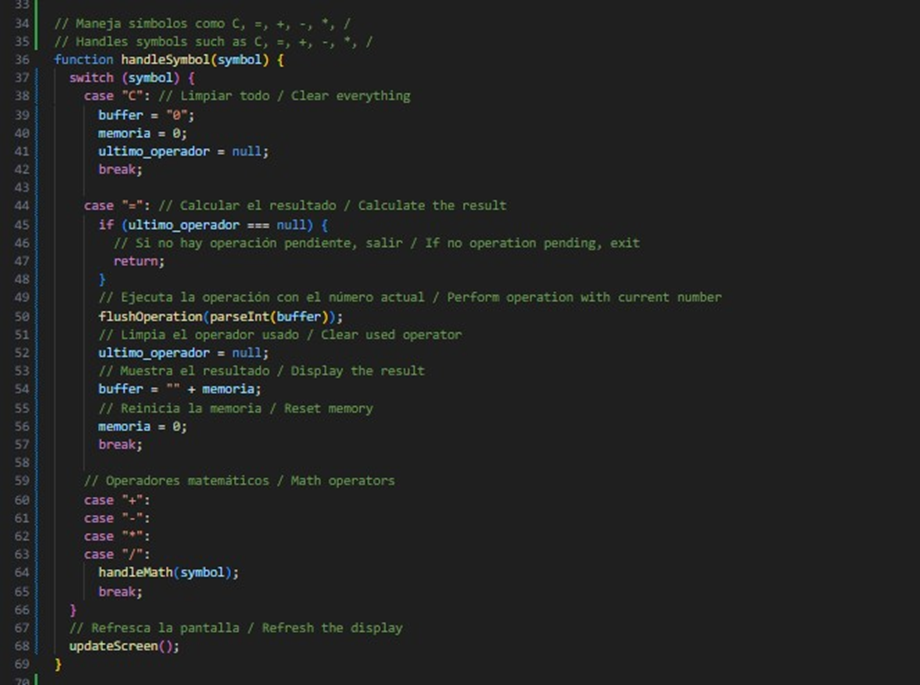
To comply with modern JavaScript best practices, it was replaced by the let declarator, which offers safer and more controlled block scope, thus ensuring better code readability and maintainability.

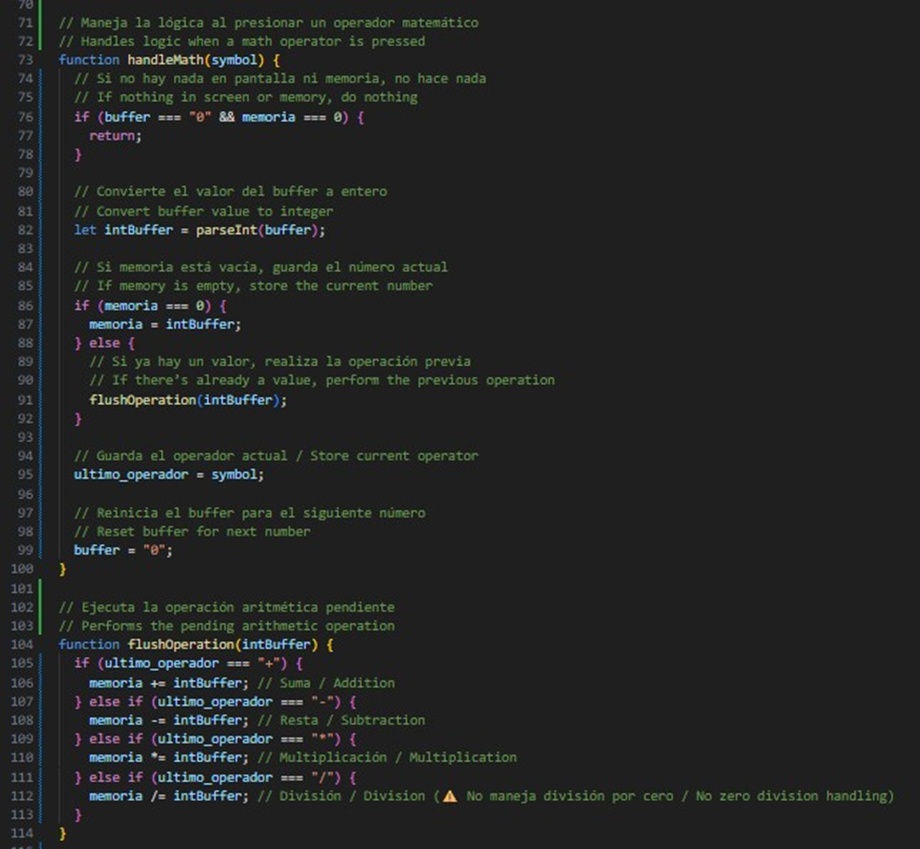


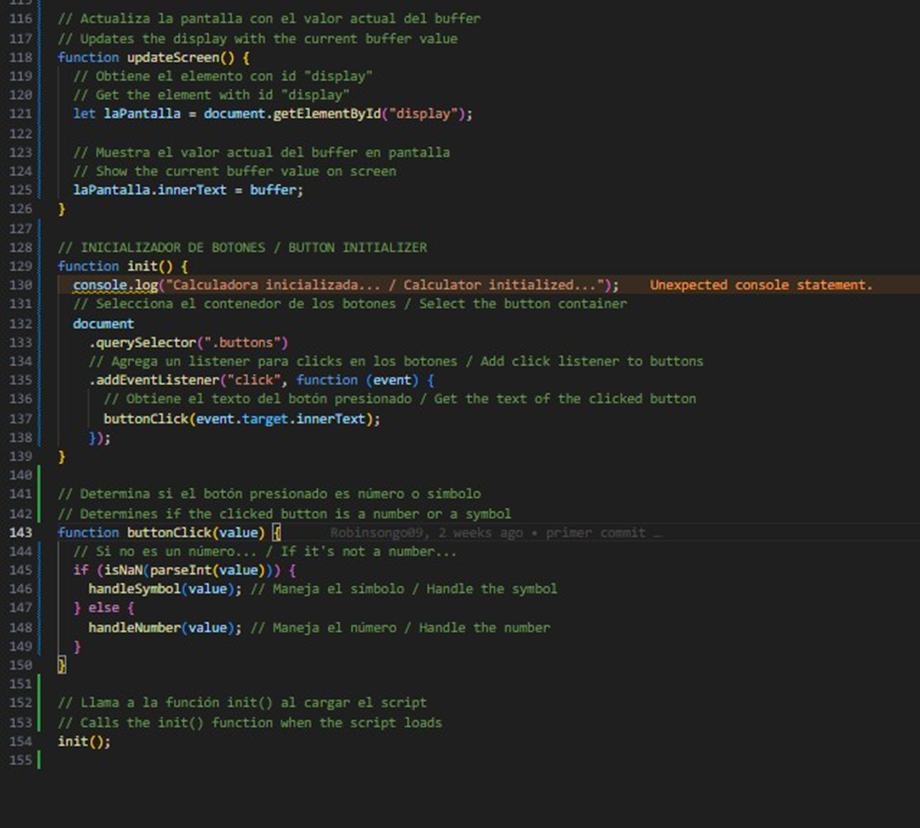
4. A continuación, se muestra el código formateado y documentado su funcionamiento.

4. Below is the formatted code and its operation documented.









REPOSITORIO : <https://github.com/Riscanevo/legado_refactoring.git>

REPOSITORY: <https://github.com/Riscanevo/legado_refactoring.git>