Proceso de Evaluación, Documentación y Herramientas de Calidad.

Bairon Sebastián Ardila Mendoza

https://github.com/BaironArd/legado-master

1. Investigación de Herramientas / Tools Investigation

ESLint:

> [Español]: ESLint es una herramienta de análisis estático utilizada para detectar errores y malas prácticas en el código JavaScript. Permite identificar desde fallos de sintaxis hasta posibles errores lógicos, además de ofrecer la posibilidad de establecer reglas personalizadas que garanticen que el proyecto cumpla con los estándares definidos por el equipo de desarrollo.
>
> [English]: ESLint is a static analysis tool used to detect errors and bad practices in JavaScript code. It helps identify syntax issues and potential logic errors, while allowing the configuration of custom rules to ensure that the project adheres to the coding standards defined by the development team.

Prettier:

> [Español]: Prettier es una herramienta de formateo automático de código que mantiene un estilo visual uniforme en todo el proyecto. Su función es ajustar la indentación, el espaciado y otros aspectos estéticos del código, lo que mejora su legibilidad y facilita el trabajo colaborativo y el mantenimiento a largo plazo.
>
> [English]: Prettier is an automatic code formatting tool that keeps a consistent visual style throughout the project. It handles indentation, spacing, and other visual aspects of the code, improving readability and making collaborative development and long-term maintenance easier.

2. Evidencia Visual / Visual Evidence

Captura de pantalla del código "Antes" / "Before" Code Screenshot:

> [Español]: Esta imagen muestra el estado inicial del código antes de aplicar las correcciones o el formateo. Se pueden observar inconsistencias en la indentación, espacios irregulares y varios avisos generados por ESLint relacionados con el uso de variables o estilo de escritura.
>
> [English]: This screenshot shows the initial state of the code before any corrections or formatting were applied. You can notice inconsistencies in indentation, irregular spacing, and several ESLint warnings related to variable usage and code styling.

```
// ------------ SCRIPT CALCULADORA LEGACY v1.2 ------------
// NO TOCAR NADA - FUNCIONA (A VECES)
var buffer = "0";
var memoria = 0;
var ultimo_operador;
function handleNumber(numStr) {
if (buffer === "0") {
buffer = numStr;
} else {
buffer += numStr;
}
updateScreen();
}
function handleSymbol(symbol) {
switch (symbol) {
case 'C':
buffer = "0";
memoria = 0;
ultimo_operador = null;
break;
case '=':
if (ultimo_operador === null) {
return;
}
flushOperation(parseInt(buffer));
ultimo_operador = null;
buffer = "" + memoria;
memoria = 0;
break;
case '+':
case '-':
case '*':
case '/':
handleMath(symbol);
break;
}
updateScreen();
}
function handleMath(symbol) {
if (buffer === '0' && memoria === 0) {
return;
}
var intBuffer = parseInt(buffer);
if (memoria === 0) {
memoria = intBuffer;
} else {
flushOperation(intBuffer);
}
ultimo_operador = symbol;
buffer = "0";
}
function flushOperation(intBuffer) {
if (ultimo_operador === '+') {
memoria += intBuffer;
} else if (ultimo_operador === '-') {
memoria -= intBuffer;
} else if (ultimo_operador === '*') {
memoria *= intBuffer;
} else if (ultimo_operador === '/') {
memoria /= intBuffer;
}
}
function updateScreen(){
var laPantalla = document.getElementById("display");
laPantalla.innerText = buffer;
}
// INICIALIZADOR DE BOTONES
function init(){
console.log("Calculadora inicializada...");
document.querySelector('.buttons').addEventListener('click', function(event) {
buttonClick(event.target.innerText);
});
}
function buttonClick(value) {
if (isNaN(parseInt(value))) {
handleSymbol(value);
} else {
handleNumber(value);
}
}
init();
```

3. Análisis de Hallazgos / Findings Analysis

Los 3 problemas más importantes encontrados con ESLint / The 3 most important issues found with ESLint:

1. [Español]: Variables declaradas pero no utilizadas

   ESLint detectó múltiples variables que habían sido definidas pero nunca empleadas dentro del programa. Este tipo de situaciones generan una mayor complejidad y pueden provocar confusión en la lectura del código. Para mantener una estructura limpia y optimizada, se eliminaron todas las variables innecesarias.

   [English]: Declared but unused variables

ESLint detected several variables that were defined but never used in the program. Such cases increase complexity and can lead to confusion when reading the code. To maintain a cleaner and more optimized structure, all unnecessary variables were removed.

2. [Español]: Inconsistencia en la indentación del código

   El código original presentaba problemas de indentación desigual en varias secciones, lo que afectaba la legibilidad y dificultaba la comprensión del flujo lógico. ESLint marcó estas inconsistencias, las cuales fueron corregidas automáticamente utilizando Prettier, garantizando una estructura uniforme y ordenada.

   [English]: Inconsistent code indentation

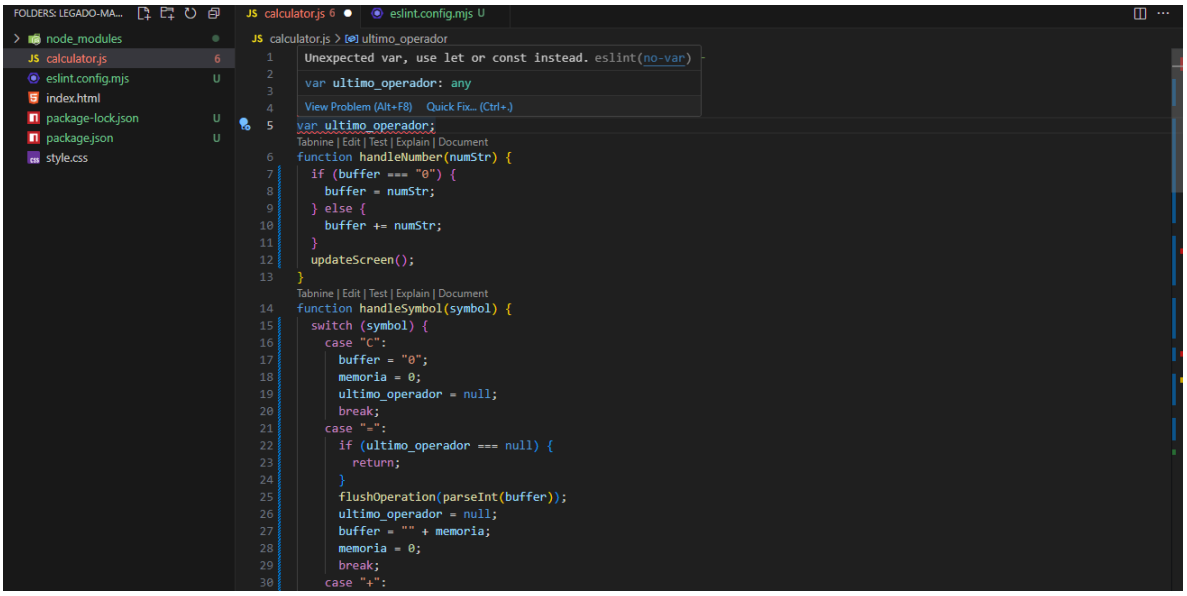
   The original code showed uneven indentation in several sections, affecting readability and making the logical flow harder to follow. ESLint flagged these inconsistencies, which were automatically fixed using Prettier, ensuring a clean and consistent structure throughout the code.

3. [Español]: Uso de “var” en lugar de “let” o “const”

   ESLint identificó el uso del tipo de variable var, una práctica obsoleta en JavaScript moderno debido a los posibles problemas de alcance (scope) que puede generar. Se reemplazó por let y const, las cuales ofrecen un control más seguro sobre las variables y son las recomendadas actualmente para garantizar un código más estable y predecible.

   [English]: Use of “var” instead of “let” or “const”

   ESLint detected the use of the var keyword, which is considered outdated in modern JavaScript because it can cause scope-related issues. It was replaced with let and const, which provide safer and more predictable variable management, following best practices for modern and clean JavaScript development.



Captura de pantalla del código "Después" / "After" Code Screenshot:

[Español]: Esta captura muestra el código una vez formateado con Prettier, antes de incluir los comentarios descriptivos. En esta etapa, el código presenta una estructura mucho más clara y uniforme, con la indentación, los saltos de línea y los espacios correctamente ajustados. El resultado es un código más limpio, legible y profesional.

[English]: This screenshot shows the code after being formatted with Prettier, but before adding the explanatory comments. At this stage, the code has a much clearer and more consistent structure, with properly adjusted indentation, line breaks, and spacing. The result is cleaner, more readable, and more professional code.

4. Documentación Bilingüe del Código / Bilingual Code Documentation

[Español]

La documentación bilingüe dentro del código se añadió con el objetivo de hacer que el proyecto sea comprensible tanto para personas que hablen español como inglés. Los comentarios fueron incluidos línea por línea, describiendo la función o finalidad de cada sección del código. Cada comentario se presenta en dos idiomas: primero en español y luego en inglés. Esta metodología facilita que cualquier desarrollador, sin importar su idioma nativo, pueda interpretar la lógica y el propósito de cada parte del programa sin dificultad.

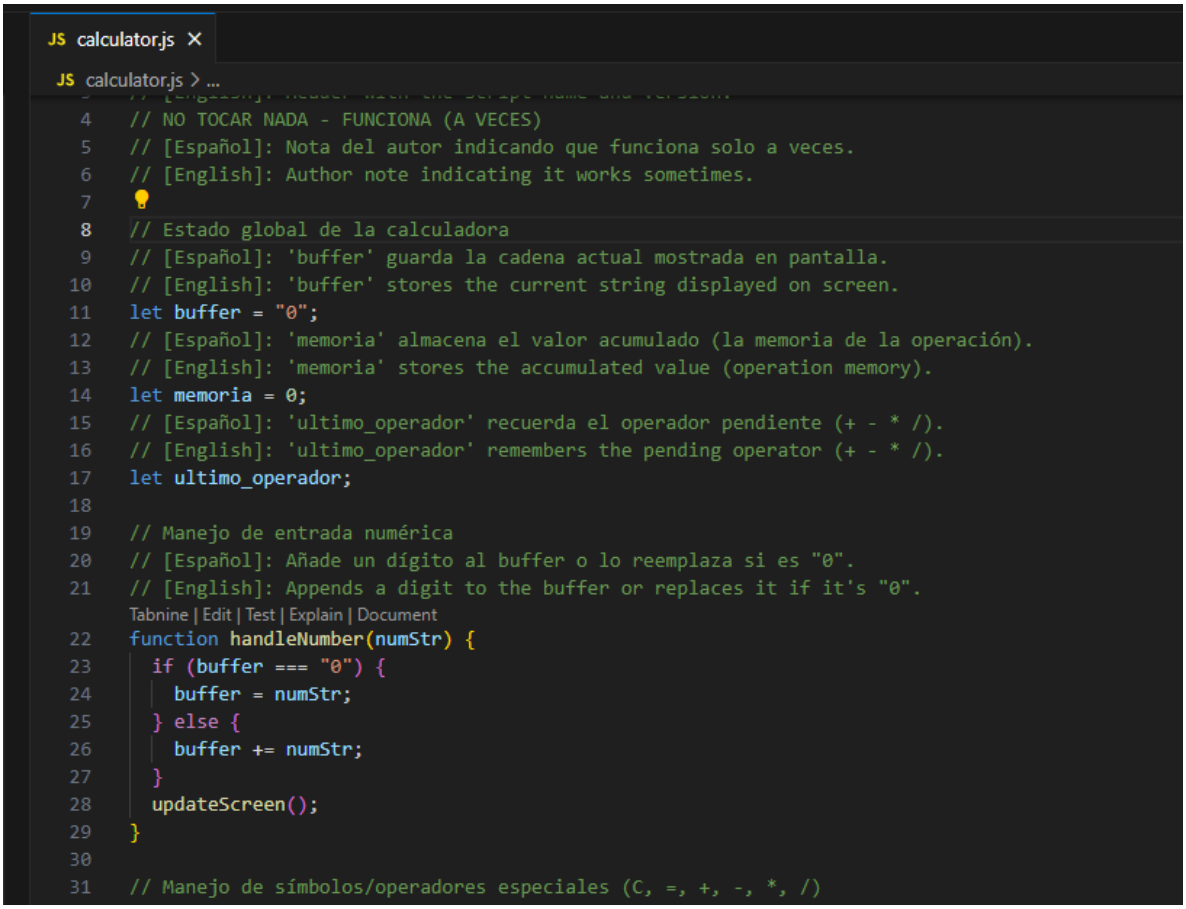
- [Español]: Se explicó con detalle cada línea o bloque relevante del código.
- [English]: Each relevant line or block of code was clearly described.

[English]

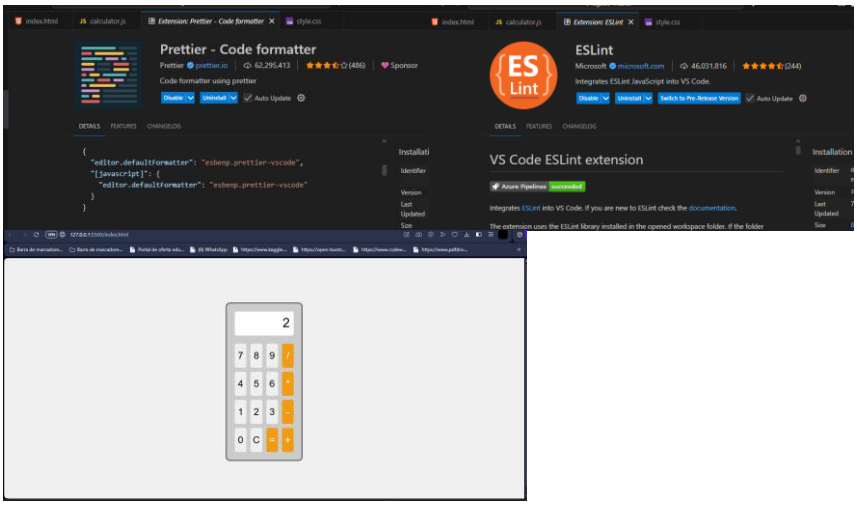
Bilingual documentation was incorporated to make the project understandable for both Spanish- and English-speaking developers. Comments were added line by line,

describing the function or intent of each section of the code. Each comment is presented in two languages: Spanish first and English second. This approach ensures that any developer, regardless of their native language, can easily follow the logic and purpose of every part of the program.

- [English]: Each relevant line or code block was clearly explained.
- [Español]: Se detalló cuidadosamente cada línea o bloque importante del código.



Anexos


Prettier - Code formatter
Prettier prettier.io 62,295,413 (486) Sponsor
Code formatter using prettier
Disable Uninstall Auto Update
DETAILS FEATURES CHANGELOG
{
"editor.defaultFormatter": "esbenp.prettier-vscode",
"[javascript]": {
"editor.defaultFormatter": "esbenp.prettier-vscode"
}
}
ESLint
Microsoft microsoft.com 46,031,816 (244)
Integrates ESLint JavaScript into VS Code.
Disable Uninstall Switch to Pre-Release Version Auto Update
DETAILS FEATURES CHANGELOG
VS Code ESLint extension
Azure Pipelines succeeded
Integrates ESLint into VS Code. If you are new to ESLint check the documentation.
2
7 8 9 /
4 5 6 *
1 2 3 -
0 C = +

Codigo full actualizado hasta el ultimo comit

```
// ------------ SCRIPT CALCULADORA LEGACY v1.2 ------------
// [Español]: Encabezado con nombre y versión del script.
// [English]: Header with the script name and version.
// NO TOCAR NADA - FUNCIONA (A VECES)
// [Español]: Nota del autor indicando que funciona solo a veces.
// [English]: Author note indicating it works sometimes.

// Estado global de la calculadora
// [Español]: 'buffer' guarda la cadena actual mostrada en pantalla.
// [English]: 'buffer' stores the current string displayed on screen.
let buffer = "0";
// [Español]: 'memoria' almacena el valor acumulado (la memoria de la operación).
// [English]: 'memoria' stores the accumulated value (operation memory).
let memoria = 0;
// [Español]: 'ultimo_operador' recuerda el operador pendiente (+ - * /).
// [English]: 'ultimo_operador' remembers the pending operator (+ - * /).
let ultimo_operador;

// Manejo de entrada numérica
// [Español]: Añade un dígito al buffer o lo reemplaza si es "0".
// [English]: Appends a digit to the buffer or replaces it if it's "0".
function handleNumber(numStr) {
  // [Español]: Si la pantalla está en "0", reemplaza con el número nuevo.
  // [English]: If the display is "0", replace it with the new number.
  if (buffer === "0") {
    buffer = numStr;
  } else {
    // [Español]: Si no, concatena el dígito al final del buffer.
    // [English]: Otherwise, concatenate the digit to the end of the buffer.
    buffer += numStr;
  }
  // [Español]: Actualiza la interfaz para mostrar el buffer actual.
  // [English]: Update the UI to show the current buffer.
  updateScreen();
}

// Manejo de símbolos/operadores especiales (C, =, +, -, *, /)
// [Español]: Decide la acción según el símbolo pulsado.
// [English]: Decide the action according to the pressed symbol.
function handleSymbol(symbol) {
  switch (symbol) {
    case "C":
      // [Español]: Limpiar todo: resetear buffer, memoria y operador.
      // [English]: Clear everything: reset buffer, memory and operator.
      buffer = "0";
      memoria = 0;
      ultimo_operador = null;
      break;
    case "=":
      // [Español]: Si no hay operador pendiente, no hace nada.
      // [English]: If there is no pending operator, do nothing.
      if (ultimo_operador === null) {
        return;
      }
      // [Español]: Ejecuta la operación pendiente usando el valor del buffer.
      // [English]: Execute the pending operation using the buffer value.
      flushOperation(parseInt(buffer));
      // [Español]: Resetea el operador y muestra el resultado en el buffer.
      // [English]: Reset the operator and show the result in the buffer.
      ultimo_operador = null;
      buffer = "" + memoria;
      memoria = 0;
      break;
    case "+":
    case "-":
    case "*":
    case "/":
      // [Español]: Si es un operador aritmético, delega a handleMath.
      // [English]: If it's an arithmetic operator, delegate to handleMath.
      handleMath(symbol);
      break;
  }
  // [Español]: Actualiza la pantalla después de procesar el símbolo.
  // [English]: Update the screen after processing the symbol.
  updateScreen();
}

// Preparar operación aritmética (cuando se pulsa un operador)
// [Español]: Mueve el número actual a memoria o aplica la operación acumulada.
// [English]: Move the current number to memory or apply the accumulated operation.
function handleMath(symbol) {
  // [Español]: Evita operaciones si no hay ningún número en buffer ni memoria.
  // [English]: Avoid operations if there is no number in buffer nor memory.
  if (buffer === "0" && memoria === 0) {
    return;
  }
  // [Español]: Convierte el buffer (cadena) a entero para operar.
  // [English]: Convert the buffer (string) to integer to operate.
  const intBuffer = parseInt(buffer);
  if (memoria === 0) {
    // [Español]: Si la memoria está vacía, asigna el valor actual.
    // [English]: If memory is empty, assign the current value.
    memoria = intBuffer;
  } else {
    // [Español]: Si ya existe un valor en memoria, ejecuta la operación pendiente.
    // [English]: If there is already a value in memory, execute the pending operation.
    flushOperation(intBuffer);
  }
  // [Español]: Guarda el operador que quedó pendiente para la siguiente entrada.
  // [English]: Store the operator that remains pending for the next input.
  ultimo_operador = symbol;
  // [Español]: Resetea el buffer para recibir el siguiente número.
  // [English]: Reset the buffer to receive the next number.
  buffer = "0";
}

// Ejecuta la operación guardada entre 'memoria' y el intBuffer actual
// [Español]: Realiza la suma, resta, multiplicación o división según el operador.
// [English]: Perform addition, subtraction, multiplication or division depending on the operator.
function flushOperation(intBuffer) {
  if (ultimo_operador === "+") {
    memoria += intBuffer;
  } else if (ultimo_operador === "-") {
    memoria -= intBuffer;
  } else if (ultimo_operador === "*") {
    memoria *= intBuffer;
  } else if (ultimo_operador === "/") {
    memoria /= intBuffer;
  }
}

// Actualiza la pantalla HTML con el valor del buffer
// [Español]: Busca el elemento con id 'display' y le asigna el texto del buffer.
// [English]: Find the element with id 'display' and set its text to the buffer.
function updateScreen() {
  const laPantalla = document.getElementById("display");
  laPantalla.innerText = buffer;
}

// INICIALIZADOR DE BOTONES
// [Español]: Añade el listener de clics al contenedor de botones para delegar eventos.
// [English]: Add click listener to the buttons container to delegate events.
function init() {
  // [Español]: Mensaje para depuración (comentado para evitar warning de console).
  // [English]: Debug message (commented out to avoid console warning).
  // console.log("Calculadora inicializada...");
  document
    .querySelector(".buttons")
    .addEventListener("click", function (event) {
      // [Español]: Llama a buttonClick pasando el texto interno del botón pulsado.
      // [English]: Call buttonClick passing the inner text of the pressed button.
      buttonClick(event.target.innerText);
    });
}

// Determina si la entrada es número o símbolo y la procesa
// [Español]: Si no es un número, se trata como símbolo; si no, como número.
// [English]: If it's not a number, treat as symbol; otherwise as number.
function buttonClick(value) {
  if (isNaN(parseInt(value))) {
    handleSymbol(value);
  } else {
    handleNumber(value);
  }
}

// Ejecuta la inicialización al cargar el script
// [Español]: Llama a init para conectar los eventos de la calculadora.
// [English]: Call init to attach the calculator events.
init();
```