Code Standardization and Debugging Workshop

https://github.com/BaironArd/legado-master/tree/entrega3legacyV3

Bairon Sebastián Ardila Mendoza- ing soft-FESC

Section 1: Debugging Links

The identified issue was found in the <link> and <script> tags. In the CSS file, the name had been written as stylse.css, adding an extra "s", when the correct name was style.css. This error was corrected. Similarly, in the JavaScript file, app-legacy.js had been referenced, omitting an "a", while the actual file name was app-legacy.js. This reference was also corrected.

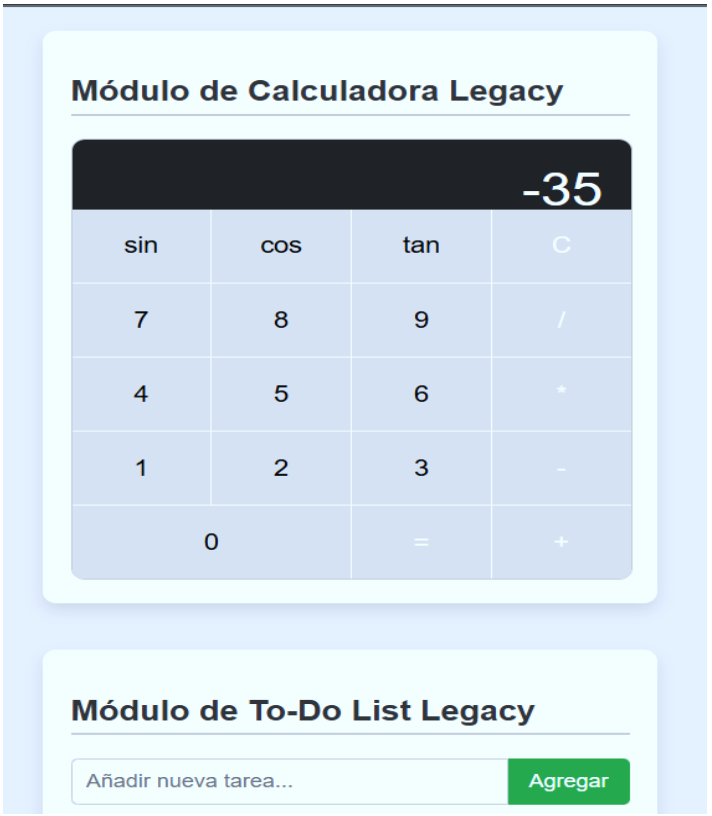
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>App "Legacy" (Auditoría)</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <div class="module-container">
        <h2>Módulo de Calculadora Legacy</h2>
        <div class="calculator">
            <div id="display">0</div>
            <div class="buttons">
                <button>sin</button>
                <button>cos</button>
                <button>tan</button>
                <button class="operator">C</button>
                <button>7</button>
                <button>8</button>
                <button>9</button>
                <button class="operator">/</button>
                <button>4</button>
                <button>5</button>
                <button>6</button>
                <button class="operator">*</button>
                <button>1</button>
                <button>2</button>
                <button>3</button>
                <button class="operator">-</button>
                <button class="zero">0</button>
                <button class="operator">=</button>
                <button class="operator">+</button>
            </div>
        </div>
    </div>

    <div class="module-container">
        <h2>Módulo de To-Do List Legacy</h2>
        <div class="todo-app">
            <div class="input-container">
                <input type="text" id="todo-input" placeholder="Añadir nueva tarea...">
                <button id="add-task-btn">Agregar</button>
            </div>
            <ul id="todo-list">
                </ul>
        </div>
    </div>

    <script src="app-legacy.js"></script>

</body>
</html>
```


Módulo de Calculadora Legacy
-35
sin
cos
tan
C
7
8
9
/
4
5
6
*
1
2
3
-
0
=
+
Módulo de To-Do List Legacy
Añadir nueva tarea...
Agregar

Section 2: Style Guidelines

This section is fundamental, as it establishes standards for all team members to follow the same format when writing code. Its purpose is to avoid disorganized styles and ensure that any member can easily understand another's work. Furthermore, it helps ensure that the project can be maintained by new developers in the future without confusion, maintaining a clear and organized structure. Finally, these rules prevent common errors arising from inconsistencies in names, formats, and data handling. In short, they function as a standard that ensures the entire team "speaks the same language" within the code.

Section 3: Standardization Process

1. "No-var" Rule

All variables declared with `var` were replaced with `let` or `const`, depending on whether their value needed to be modified. This was done because `var` presents problems with scope and hoisting, which can lead to logical errors. Instead, we consistently limit the scope to the block and prevent accidental redefinitions.

Benefits:

• Greater clarity and control over variable behavior.

• Prevention of unexpected effects outside the block where it is declared.

• Facilitates maintenance in collaborative environments.

2. CamelCase Rule

Variables written in Snake_case format were modified to adapt to camelCase, the JavaScript standard. This ensures uniformity in naming variables, functions, and properties, improving code readability and maintainability.

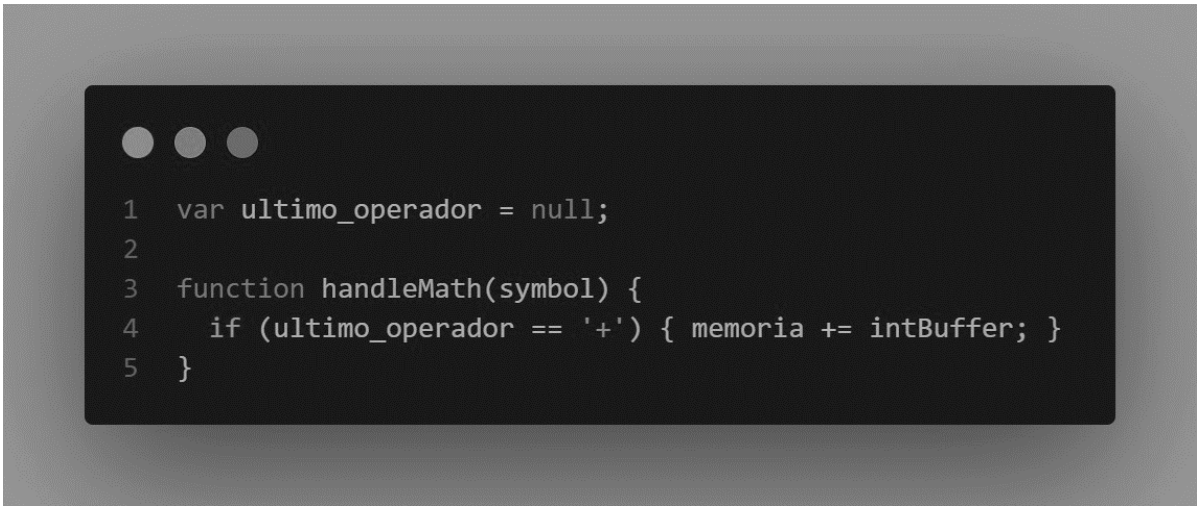
Benefits:

• Greater visual consistency.

• Simplifies searching and refactoring in large projects.

• Complies with standards such as ECMAScript and the Airbnb Style Guide.

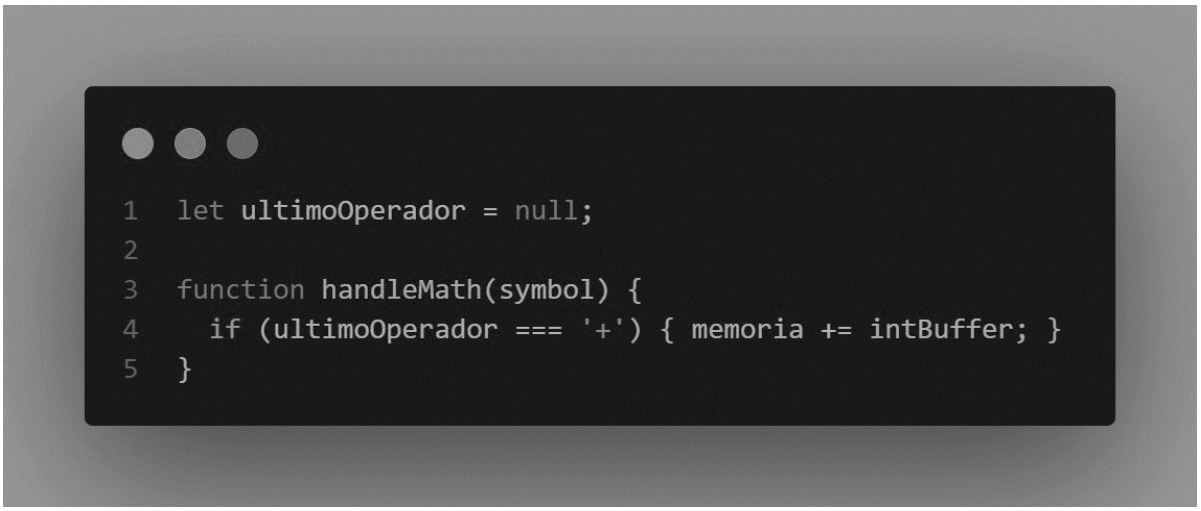
Previously used with snake_case



```
var ultimo_operador = null;

function handleMath(symbol) {
  if (ultimo_operador == '+') { memoria += intBuffer; }
}
```

Now used with camelCase



```
let ultimoOperador = null;

function handleMath(symbol) {
  if (ultimoOperador === '+') { memoria += intBuffer; }
}
```

3. The “eqeqeq” Rule (Strict Equality)

All the operators == and != were replaced with === and !==. Loose operators can perform implicit type conversions, leading to incorrect comparisons (e.g., '0' == 0). Strict operators compare both value and type, preventing silent errors.

Benefits:

• Prevents unexpected results due to automatic type coercion.

• Increases the reliability of comparisons.

• Facilitates error detection during the testing process.

Before

```
// MODULO DE CALCULADORA v2 - LEGACY
var buffer = "0";
var memoria = 0;
var ultimo_operador;
var historial = [];
const MAX_HISTORY_ITEMS = 5; // El dev anterior al menos puso una const

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; }
      flushOperationAndLog(parseInt(buffer));
      ultimo_operador = null;
      buffer = "" + memoria;
      memoria = 0;
      break;
    case '+': case '-': case '*': case '/':
      handleMath(symbol);
      break;
    case 'sin': case 'cos': case 'tan':
      if (buffer == "0") return;
      var cientifico_result;
      var val = parseFloat(buffer);
      if (symbol == 'sin') { cientifico_result = Math.sin(val); }
      else if (symbol == 'cos') { cientifico_result = Math.cos(val); }
      else if (symbol == 'tan') { cientifico_result = Math.tan(val); }
      buffer = "" + cientifico_result;
      var logEntry = symbol + "(" + val + ") = " + cientifico_result;
      logHistory(logEntry);
      break;
  }
  updateScreen();
}

function handleMath(symbol) {
  if (buffer == '0' && memoria == 0) { return; }
  var intBuffer = parseInt(buffer);
  if (memoria == 0) {
    memoria = intBuffer;
  } else {
    flushOperationAndLog(intBuffer);
  }
  ultimo_operador = symbol;
  buffer = "0";
}

function flushOperationAndLog(intBuffer) {
  var operacionPrevia = ultimo_operador;
  var memoriaPrevia = memoria;

  if (ultimo_operador == '+') { memoria += intBuffer; }
  else if (ultimo_operador == '-') { memoria -= intBuffer; }
  else if (ultimo_operador == '*') { memoria *= intBuffer; }
  else if (ultimo_operador == '/') { memoria /= intBuffer; }

  var logEntry = memoriaPrevia + " " + operacionPrevia + " " + intBuffer + " = " + memoria;
  logHistory(logEntry);
}

function logHistory(logEntry) {
    historial.push(logEntry);
    if (historial.length > MAX_HISTORY_ITEMS) {
        historial.shift();
    }
    console.log(historial);
}

function updateScreen(){
  // Error común: El ID "display" no existe en el HTML del To-Do.
  // Esto es un error de lógica, pero hoy nos enfocamos en ESTILO.
  var display_element = document.getElementById("display");
  if (display_element != null) {
      display_element.innerText = buffer;
  }
}

function init_calculadora(){
  var calculator_buttons = document.querySelector(".buttons");
  if (calculator_buttons != null) {
      calculator_buttons.addEventListener('click', function(event) {
        buttonClick(event.target.innerText);
      });
  }
}
function buttonClick(value) {
  if (isNaN(parseInt(value))) { handleSymbol(value); } else { handleNumber(value); }
}
```

```javascript
// ------------------------------------------------------------
// MODULO DE TO-DO LIST (en el mismo archivo... ¡qué horror!)
// ------------------------------------------------------------

var todo_list = []; // Lista de tareas
var user_name = "Default User"; // Otro var

function agregar_tarea() {
    var input_element = document.getElementById("todo-input");
    var texto_tarea = input_element.value;

    if (texto_tarea == "") {
        alert("Error: La tarea no puede estar vacía.");
        return;
    }

    // Chequeo de duplicados (escrito con 'var' y '==')
    var duplicado = false;
    for (var i = 0; i < todo_list.length; i++) {
        if (todo_list[i].texto == texto_tarea) {
            duplicado = true;
            break;
        }
    }

    if (duplicado == true) {
        alert("Error: Tarea duplicada.");
        return;
    }

    var nueva_tarea = {
        "id": Date.now(),
        "texto": texto_tarea,
        "completada": false
    };

    todo_list.push(nueva_tarea);
    input_element.value = ""; // Limpiar el input
    dibujar_lista_tareas();
}

function dibujar_lista_tareas() {
    var lista_html = document.getElementById("todo-list");
    if (lista_html == null) return; // Salir si no estamos en la página del To-Do

    lista_html.innerHTML = ""; // Limpiar la lista

    for (var i = 0; i < todo_list.length; i++) {
        var tarea_actual = todo_list[i];
        var elemento_lista = document.createElement("li");

        elemento_lista.innerText = tarea_actual.texto;

        if (tarea_actual.completada == true) {
            elemento_lista.style.textDecoration = "line-through";
        }

        // Evento para borrar (mala práctica, pero la dejamos por ahora)
        elemento_lista.addEventListener("click", function() {
            // Esto es complejo, pero nos enfocamos en el estilo
            // (Encontrar el índice y borrarlo)
        });

        lista_html.appendChild(elemento_lista);
    }
}

function init_todo_list(){
    var boton_agregar = document.getElementById("add-task-btn");
    if (boton_agregar != null) {
        boton_agregar.addEventListener("click", agregar_tarea);
    }
    dibujar_lista_tareas(); // Dibujar al cargar
}

// Inicializar AMBOS módulos
init_calculadora();
init_todo_list();
```

After

```
// MODULO DE CALCULADORA v2 - LEGACY CORREGIDO
let buffer = '0';
let memoria = 0;
let ultimoOperador;
const historial = [];
const MAX_HISTORY_ITEMS = 5;

function handleNumber(numStr) {
  if (buffer === '0') {
    buffer = numStr;
  } else {
    buffer += numStr;
  }
  updateScreen();
}

function handleSymbol(symbol) {
  switch (symbol) {
    case 'C':
      buffer = '0';
      memoria = 0;
      ultimoOperador = null;
      break;
    case '=':
      if (ultimoOperador === null) {
        return;
      }
      flushOperationAndLog(parseInt(buffer));
      ultimoOperador = null;
      buffer = '' + memoria;
      memoria = 0;
      break;
    case '+':
    case '-':
    case '*':
    case '/':
      handleMath(symbol);
      break;
    case 'sin':
    case 'cos':
    case 'tan':
      if (buffer === '0') return;
      let cientificoResult;
      const val = parseFloat(buffer);
      if (symbol === 'sin') {
        cientificoResult = Math.sin(val);
      } else if (symbol === 'cos') {
        cientificoResult = Math.cos(val);
      } else if (symbol === 'tan') {
        cientificoResult = Math.tan(val);
      }
      buffer = '' + cientificoResult;
      const logEntryCientifico = `${symbol}(${val}) = ${cientificoResult}`;
      logHistory(logEntryCientifico);
      break;
  }
  updateScreen();
}

function handleMath(symbol) {
  if (buffer === '0' && memoria === 0) {
    return;
  }
  const intBuffer = parseInt(buffer);
  if (memoria === 0) {
    memoria = intBuffer;
  } else {
    flushOperationAndLog(intBuffer);
  }
  ultimoOperador = symbol;
  buffer = '0';
}

function flushOperationAndLog(intBuffer) {
  const operacionPrevia = ultimoOperador;
  const memoriaPrevia = memoria;

  if (ultimoOperador === '+') {
    memoria += intBuffer;
  } else if (ultimoOperador === '-') {
    memoria -= intBuffer;
  } else if (ultimoOperador === '*') {
    memoria *= intBuffer;
  } else if (ultimoOperador === '/') {
    memoria /= intBuffer;
  }

  const logEntry = `${memoriaPrevia} ${operacionPrevia} ${intBuffer} = ${memoria}`;
  logHistory(logEntry);
}

function logHistory(logEntry) {
  historial.push(logEntry);
  if (historial.length > MAX_HISTORY_ITEMS) {
    historial.shift();
  }
  console.log(historial);
}

function updateScreen() {
  const displayElement = document.getElementById('display');
  if (displayElement !== null) {
    displayElement.innerText = buffer;
  }
}

function initCalculadora() {
  const calculatorButtons = document.querySelector('.buttons');
  if (calculatorButtons !== null) {
    calculatorButtons.addEventListener('click', function (event) {
      buttonClick(event.target.innerText);
    });
  }
}

function buttonClick(value) {
  if (isNaN(parseInt(value))) {
    handleSymbol(value);
  } else {
    handleNumber(value);
  }
}
```

```javascript
// ------------------------------------------------------------
// MODULO DE TO-DO LIST CORREGIDO
// ------------------------------------------------------------

let todoList = [];
let userName = 'Default User';

function agregarTarea() {
  const inputElement = document.getElementById('todo-input');
  const textoTarea = inputElement.value;

  if (textoTarea === '') {
    alert('Error: La tarea no puede estar vacía.');
    return;
  }

  let duplicado = false;
  for (let i = 0; i < todoList.length; i++) {
    if (todoList[i].texto === textoTarea) {
      duplicado = true;
      break;
    }
  }

  if (duplicado === true) {
    alert('Error: Tarea duplicada.');
    return;
  }

  const nuevaTarea = {
    id: Date.now(),
    texto: textoTarea,
    completada: false
  };

  todoList.push(nuevaTarea);
  inputElement.value = '';
  dibujarListaTareas();
}

function dibujarListaTareas() {
  const listaHtml = document.getElementById('todo-list');
  if (listaHtml === null) return;

  listaHtml.innerHTML = '';

  for (let i = 0; i < todoList.length; i++) {
    const tareaActual = todoList[i];
    const elementoLista = document.createElement('li');

    elementoLista.innerText = tareaActual.texto;

    if (tareaActual.completada === true) {
      elementoLista.style.textDecoration = 'line-through';
    }

    elementoLista.addEventListener('click', function () {
    });

    listaHtml.appendChild(elementoLista);
  }
}

function initTodoList() {
  const botonAgregar = document.getElementById('add-task-btn');
  if (botonAgregar !== null) {
    botonAgregar.addEventListener('click', agregarTarea);
  }
  dibujarListaTareas();

initCalculadora();
initTodoList();
```