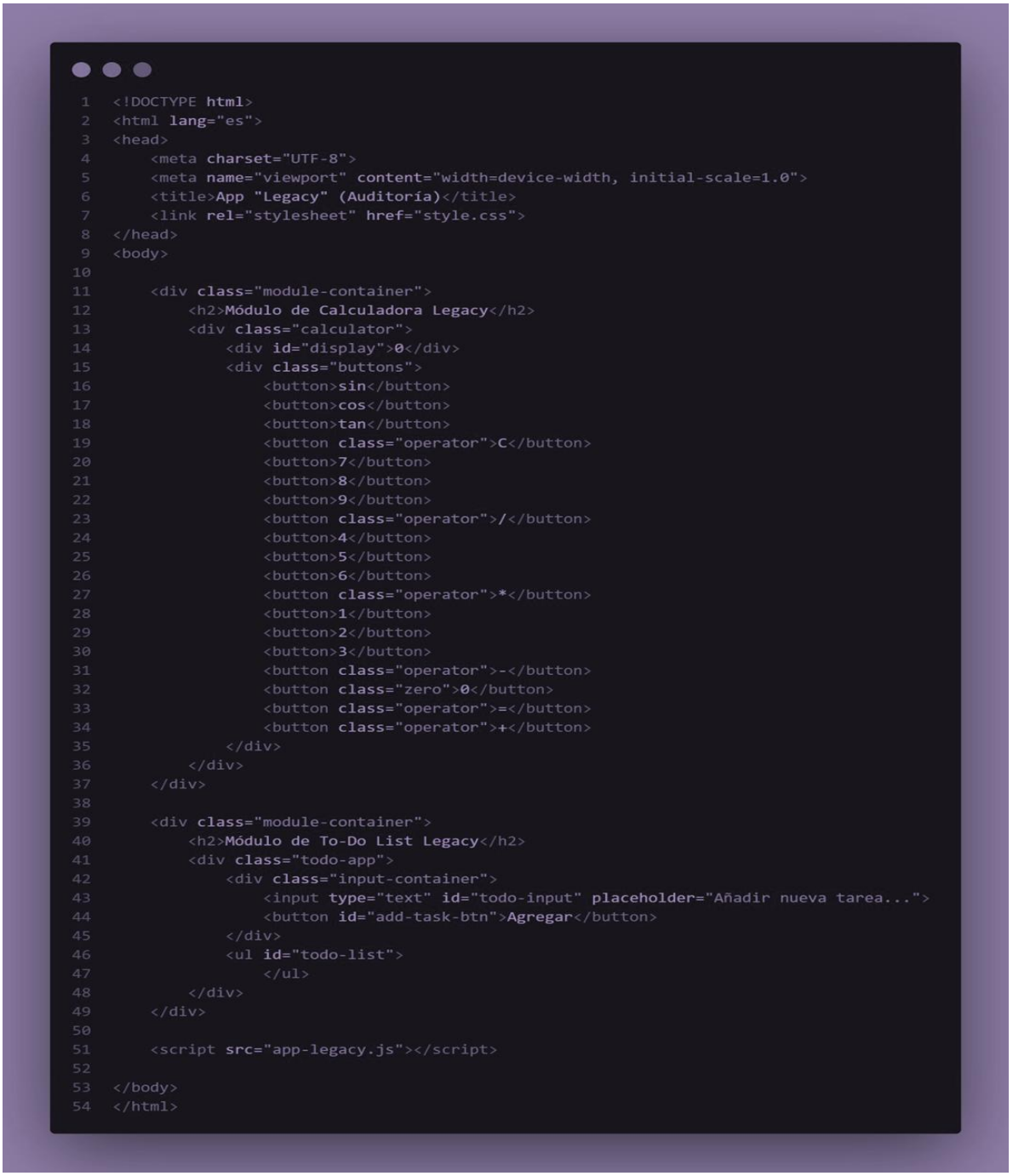
Taller de Estandarización y Depuración de Código

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

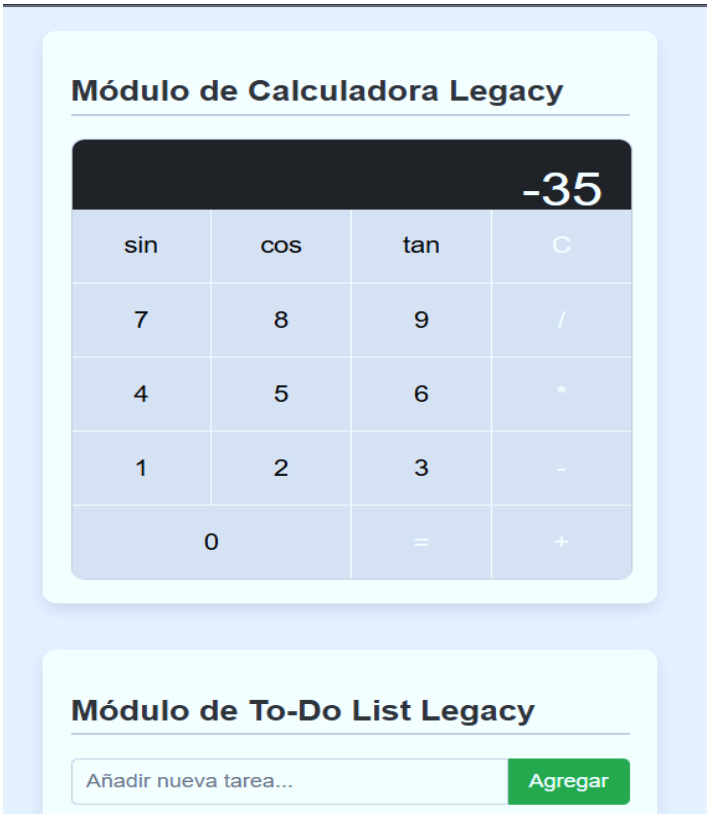
Bairon Sebastián Ardila Mendoza- ing soft-FESC

Sección 1: Depuración de Enlaces

El inconveniente identificado se encontraba en las etiquetas <link> y <script>. En el archivo CSS, el nombre había sido escrito como stylse.css, añadiendo una “s” extra, cuando el nombre correcto era style.css. Este error fue corregido. De igual manera, en el archivo JavaScript se había referenciado app-legcy.js, omitiendo una “a”, mientras que el nombre real del archivo era app-legacy.js. Esta referencia también fue ajustada correctamente.



```html
<!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

Sección 2: Guías de Estilo

Esta sección es fundamental, ya que establece normas para que todos los integrantes del equipo sigan un mismo formato al escribir código. Su propósito es evitar estilos desorganizados y garantizar que cualquier miembro pueda comprender el trabajo de otro sin dificultad. Además, contribuye a que el proyecto pueda ser mantenido por nuevos desarrolladores en el futuro sin generar confusión, manteniendo una estructura clara y ordenada. Finalmente, estas reglas previenen errores comunes derivados de inconsistencias en nombres, formatos y manejo de datos. En resumen, funcionan como un estándar que asegura que todo el equipo "hable el mismo idioma" dentro del código.

Sección 3: Proceso de Estandarización

1. Regla "no-var"

Todas las variables declaradas con `var` fueron sustituidas por `let` o `const`, dependiendo de si su valor necesitaba o no ser modificado. Esto se realizó debido a que `var` presenta problemas con el alcance (scope) y el hoisting, lo que puede generar errores lógicos. En cambio, `let` y `const` restringen el alcance al bloque y previenen redefiniciones accidentales.

Beneficios:

- Mayor claridad y control del comportamiento de las variables.
- Prevención de efectos inesperados fuera del bloque donde se declaran.
- Facilita el mantenimiento en entornos colaborativos.

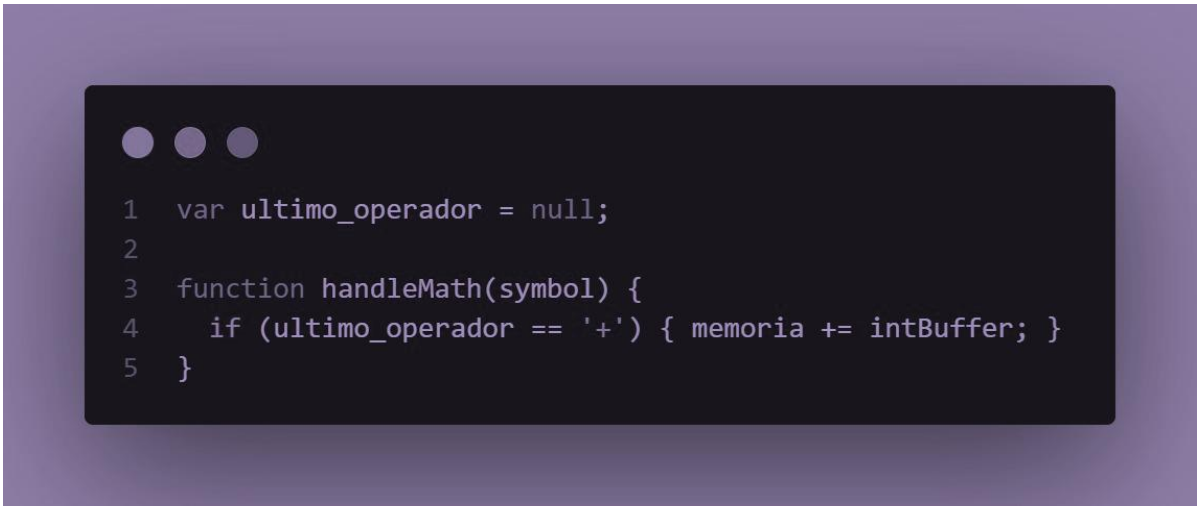
2. Regla "camelCase"

Se modificaron las variables escritas en formato `snake_case` para adaptarlas al estilo `camelCase`, que es el estándar en JavaScript. Esto asegura uniformidad al nombrar variables, funciones y propiedades, mejorando la lectura y mantenimiento del código.

Beneficios:

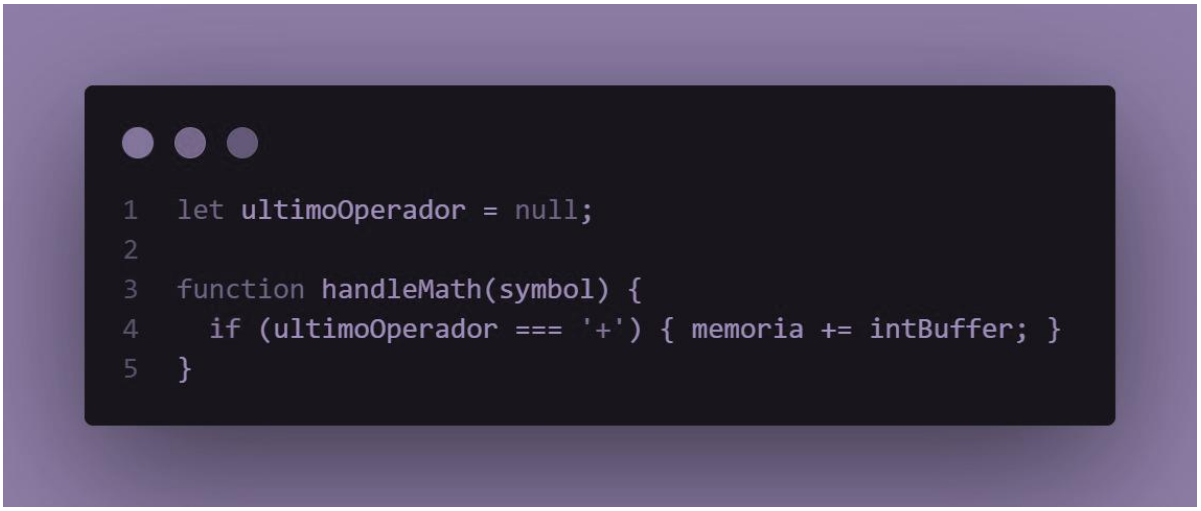
- Mayor consistencia visual.
- Simplifica la búsqueda y refactorización en proyectos grandes.
- Cumple con estándares como ECMAScript y Airbnb Style Guide.

Antes con snake_case



```
var ultimo_operador = null;

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

Ahora con camelCase



```
let ultimoOperador = null;

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

3. Regla “eqeqeq” (igualdad estricta)

Se reemplazaron todos los operadores `==` y `!=` por `===` y `!==`. Los operadores laxos pueden realizar conversiones implícitas de tipo, provocando comparaciones incorrectas (por ejemplo: `'0' == 0`). Los operadores estrictos comparan tanto valor como tipo, evitando errores silenciosos.

Beneficios:

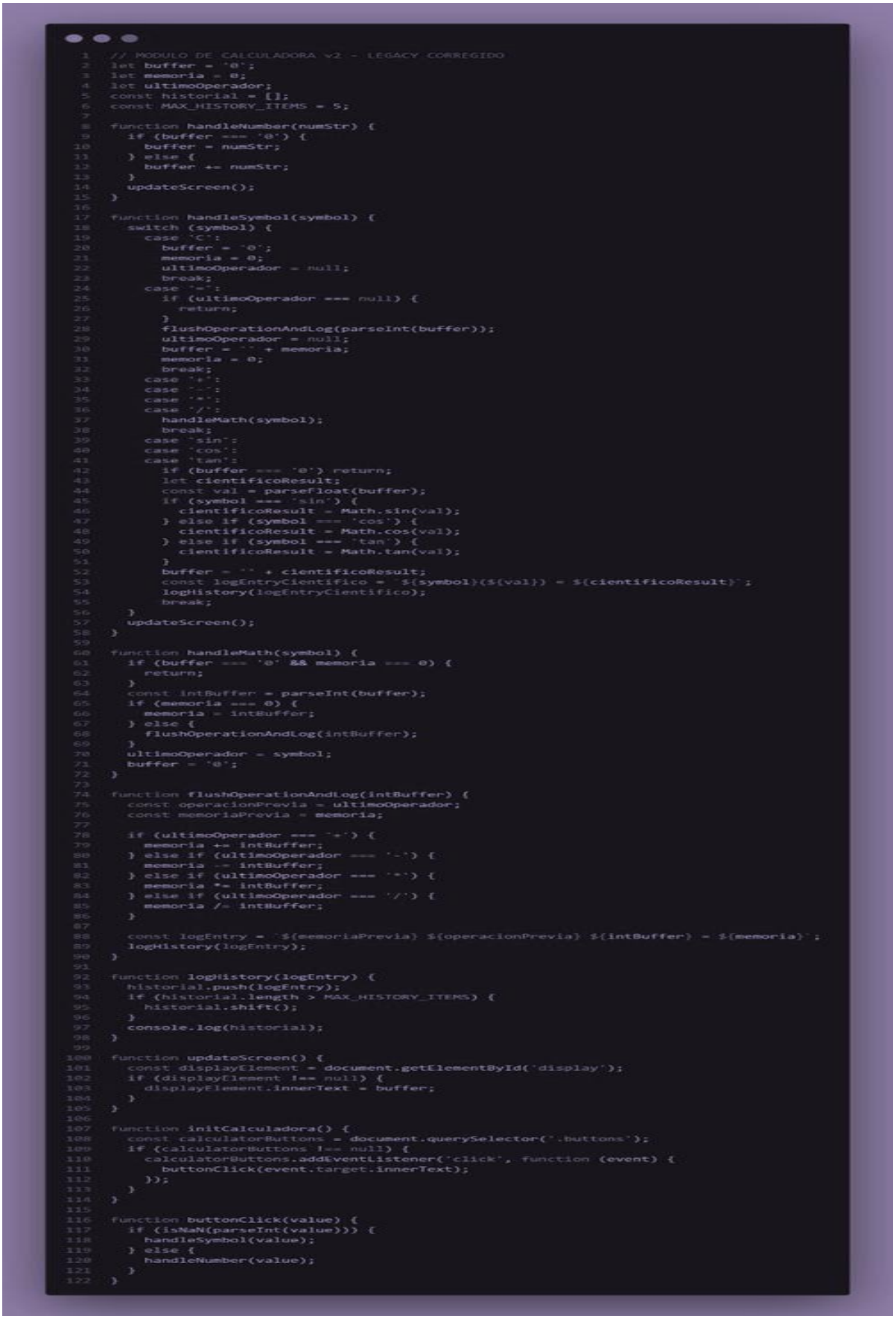
- Evita resultados inesperados por coerción automática de tipos.
- Aumenta la fiabilidad en las comparaciones.
- Facilita la detección de errores durante el proceso de pruebas.

Antes

```
// 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();
```

Después



```
// 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);
  }
}
```

```
// ------------------------------------------------------------
// 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();
```