**Отчет по Лабораторной работе 2: Добавление JavaScript Кода и Функциональности**

**Выполнил студент:** Ованнисян Петрос Абгарович

**Группа:** СГН3-64Б

**Обзор:**

Данный отчет посвящен анализу JavaScript-кода, добавленного в файл calculator.html для обеспечения функциональности калькулятора. Целью является понимание логики работы функций, обработки событий и взаимодействия с HTML-элементами.

**JavaScript Код:**

JavaScript-код отвечает за следующие основные задачи:

• Управление отображением: Отображение введенных чисел, операторов и результатов вычислений в текстовом поле.

• Обработка нажатий кнопок: Реакция на нажатия кнопок (цифры, операторы, AC, =).

• Выполнение вычислений: Выполнение математических операций на основе введенных данных.

• Обработка ошибок: Обработка ситуаций, когда вычисление не может быть выполнено (например, деление на ноль).

**Основные функции JavaScript:**

• let displayValue = '0';: Инициализация переменной, хранящей текущее значение на экране калькулятора.

• function updateDisplay() { ... }: Функция, обновляющая текстовое поле (<input type="text" id="display" ...>) текущим значением displayValue.

• function appendNumber(number) { ... }: Функция, добавляющая цифру (number) к текущему значению на экране. Если текущее значение равно '0', оно заменяется на введенную цифру.

• function appendOperator(operator) { ... }: Функция, добавляющая оператор (operator) (+, -, \\*, /) к текущему значению на экране.

• function clearDisplay() { ... }: Функция, очищающая экран калькулятора, устанавливая displayValue в '0'.

• function calculate() { ... }: Функция, выполняющая вычисление на основе текущего значения displayValue.

\* eval(displayValue): Использует функцию eval() для вычисления математического выражения, содержащегося в displayValue. Внимание: eval() представляет угрозу безопасности и не рекомендуется к использованию в production-приложениях.

\* try...catch: Блок try...catch используется для обработки ошибок, возникающих при вычислении выражения (например, при делении на ноль). Если возникает ошибка, displayValue устанавливается в 'Error'.

**Обработка событий:**

Атрибут onclick в каждой кнопке (<button onclick="...">) используется для вызова соответствующей JavaScript-функции при нажатии на кнопку. Например:

• <button onclick="appendNumber('7')">7</button>: При нажатии на кнопку "7" вызывается функция appendNumber('7').

• <button onclick="calculate()" class="span-two">=</button>: При нажатии на кнопку "=" вызывается функция calculate().

• <button onclick="clearDisplay()" class="span-two">AC</button>: При нажатии на кнопку "AC" вызывается функция clearDisplay().

**Анализ и Выводы:**

• Код JavaScript достаточно прост и понятен, что облегчает понимание ло

гики работы калькулятора.

• Использование eval() упрощает выполнение вычислений, но представляет угрозу безопасности.

• Обработка ошибок с помощью try...catch обеспечивает устойчивость работы калькулятора.

• Функции хорошо структурированы и выполняют конкретные задачи.

**Рекомендации:**

• Заменить eval() на более безопасный метод вычисления выражений. Рассмотреть использование библиотеки math.js или разработку собственного парсера выражений.

• Добавить проверку ввода, чтобы предотвратить ввод некорректных данных (например, несколько операторов подряд).

• Улучшить обработку ошибок, предоставляя более информативные сообщения об ошибках.

• Реализовать более сложные функции, такие как память, проценты, возведение в степень и т.д.

• Использовать слушатели событий (event listeners) вместо атрибутов onclick для разделения HTML и JavaScript. Это сделает код более гибким и управляемым. Например: document.getElementById("button7").addEventListener("click", function() { appendNumber('7'); });

• Рассмотреть возможность модульной структуры кода для более удобной организации и поддержки, особенно если планируется дальнейшее развитие функциональности.