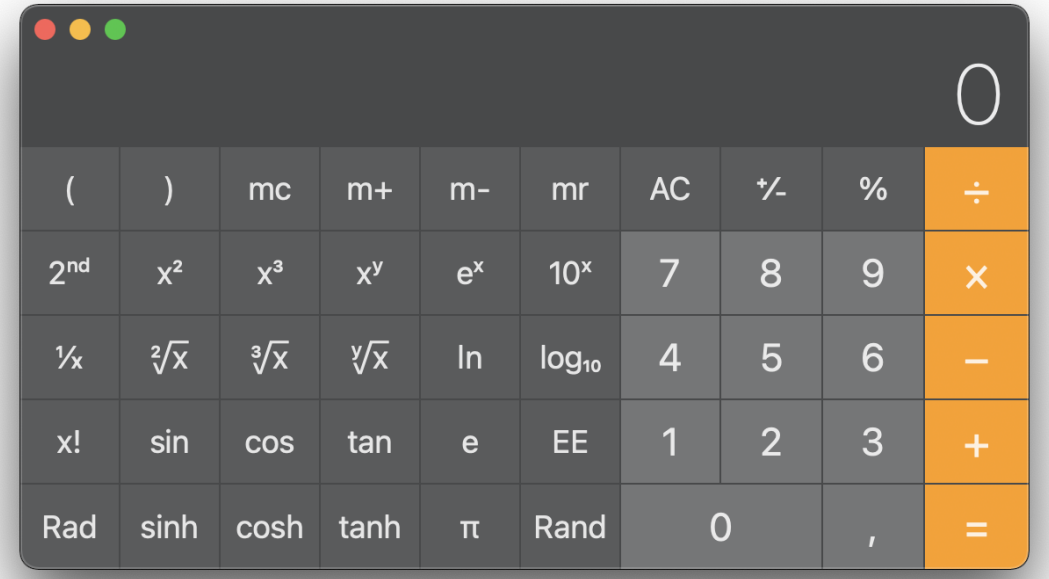
Level 0: Custom Calculator

Экосистема:

Требуется инициализировать базовое JS приложение (без использования каких-либо библиотек и фреймворков). Для этого приложения создать в том числе Webpack конфиг, который позволяет запускать это приложение с оптимизированным кодом. Webpack-конфиг сделать с оптимизации кода. То есть на выходе в билде должно получиться 2 файла: HTML и JS. Вне зависимости от того, сколько файлов было на этапе разработки.

Суть приложения: реализовать свой калькулятор с проверкой всех условий. Пример дизайна калькулятора можно увидеть ниже.



Функциональные требования:

1. Из математических функций необходимо реализовать: деление, умножение, вычитание, сложение, процент, смена знака, квадрат, куб, x в степени y, 10 в степени x, 1 / x, коренеь квадратный, корень кубический, корень степени y, факториал.

2. Скобки реализовывать не надо.

3. Надо реализовать MC, M+, M-, MR.

4. При выполнении операций реализовать следующее поведение: когда введен левый операнд, правый операнд, и операция - при нажатии на следующую операцию или “=” текущая операция должна исполниться и поставиться в левый операнд.

Технические требования:

5. Использование фреймворков не требуется. Писать надо на чистых JS-файлах. Использование JQuery тоже не разрешено. Файлы разбивать по соответствующим группам.

6. При реализации изучить и применить паттерн Command. Внедрить его в проект на обработку нажатий операций и цифр.

7. **Нельзя использовать eval** для обработки функций. Не используем Mat, встроенный в JS, не используем никаких сторонних библиотек для вычислений (вроде Mathjs).

8. Все исключения должны быть обработаны. При делении на ноль в любом возможном случае (в том числе и из выгрузки из памяти) должна быть обработка.

9. Перед разработкой надо настроить ESLint для приложения. В качестве дополнения можно настроить Prettier и связать его с ESLint. *Будет дополнительным плюсом реализовать pre-commit hook, который не позволит сделать push в репозиторий, если в приложении присутствуют ошибки ESLint.*

10. Написание документации к проекту; в README-файле проекта перед сдачей задания должна быть написана краткая документация на английском или русском, состоящая из 4 пунктов: “Task” (ссылка на этот документ), “How to run the app” (инструкции по билду приложения), *плюсом будет написание краткого описания структуры папок: для каждой папки описать, какого типа файлы она хранит.*

11. Надо писать Unit-тесты используя Jest. Протестировать надо все функции для вычисления математических операций.

Дополнительные баллы можно получить за:

1. Действительно удобный пользовательский интерфейс.

2. Реализованный Theme-менеджмент. Возможность с легкостью поменять цветовую схему приложения **из приложения** (надо реализовать переключатели для пользователя).

3. Реализация операции отмены предыдущего действия (паттерн Snapshot или Command).