Calculator App

This is a simple calculator app built using React. The app allows users to perform basic arithmetic operations like addition, subtraction, multiplication, and division.

Components

App.js

The main component that holds the state of the calculator and renders the UI. It uses the useReducer hook to manage state updates based on dispatched actions.

State

* currentOperand: The current operand being entered or displayed.
* overwrite: A flag indicating whether the current operand should overwrite the existing value.
* previousOperand: The operand from the previous calculation.
* operation: The selected arithmetic operation.

Actions

This section defines a set of action types that can be dispatched by the reducer. Actions are events that trigger state updates in your application. Each action has a type, which is a string identifier describing the action.

The app defines several actions to handle different user interactions:

* ADD\_DIGIT: Adds a digit to the current operand.
* DELETE\_DIGIT: Deletes the last digit from the current operand.
* CHOOSE\_OPERATION: Selects an arithmetic operation.
* CLEAR: Clears the calculator state.
* EVALUATE: Evaluates the expression and updates the state.

Reducer

The reducer is a pure function responsible for updating the state based on the dispatched action. It takes the current state and an action as parameters and returns a new state.

The reducer function specifies how the state should be updated based on the dispatched actions. It includes logic for adding digits, choosing operations, clearing the state, and evaluating expressions.

* ADD\_DIGIT Action: This case in the reducer handles the ADD\_DIGIT action. It checks whether the calculator is in "overwrite" mode and adjusts the state accordingly. It also includes additional checks to handle special cases, such as preventing multiple leading zeros or redundant decimal points.
* DELETE\_DIGIT Action: This case in the reducer handles the DELETE\_DIGIT action. It checks whether the calculator is in "overwrite" mode and adjusts the state accordingly. It also handles cases where there is only one digit left, resetting the current operand to "null."
* CHOOSE\_OPERATION Action: This case in the reducer handles the CHOOSE\_OPERATION action. It checks various conditions before updating the state, such as ensuring there is a current operand or a previous operand. It sets the operation and may perform additional adjustments to the state.
* CLEAR Action: This case in the reducer handles the CLEAR action. It resets the calculator state to its initial values, clearing the current operand, previous operand, and operation.
* EVALUATE Action: This case in the reducer handles the EVALUATE action. It checks whether the necessary conditions are met for evaluating the expression and then calculates the result using the evaluate function. It sets the new state with the calculated result.

Evaluation Logic

The evaluate function performs arithmetic operations based on the selected operation. The evaluate function takes the current, previous operands, and the operation to perform arithmetic calculations. It checks for valid numeric values, performs the specified operation, and returns the computed result.

Formatting

The formatOperand function takes an operand and formats it for display. It handles cases where the operand is null and splits the operand into integer and decimal parts for proper formatting.

DigitButton.js

A functional component representing a button for entering digits. It dispatches the ADD\_DIGIT action when clicked.

OperationButton.js

A functional component representing a button for selecting arithmetic operations. It dispatches the CHOOSE\_OPERATION action when clicked.

How to Run

To run the calculator app, follow these steps:

* Clone the repository to your local machine.
* Navigate to the project directory.
* Run npm install to install dependencies.
* Run npm start to start the development server.
* Open your browser and go to http://localhost:3000 to view the calculator app.