Calculator program using ReatJS

Calculator.jsx:

```
import React, { useState } from 'react';

const Calculator = () => {
  const [input, setInput] = useState('');
  const [result, setResult] = useState('');

  // Handles button click
  const handleClick = (value) => {
    if (value === '=') {
      try {
        // Safely evaluate the mathematical expression
        const calcResult = eval(input); // Note: Avoid `eval` in production, use libraries like
math.js
        setResult(calcResult);
      } catch (error) {
        setResult('Error');
      }
    } else if (value === 'C') {
      setInput('');
      setResult('');
    } else {
      setInput(input + value);
    }
  };

  return (
    <div style={calculatorStyle}>
      <h1>React Calculator</h1>
      <div style={displayStyle}>
        <div style={inputStyle}>{input}</div>
        <div style={resultStyle}>{result}</div>
      </div>
      <div style={buttonContainerStyle}>
        {['7', '8', '9', '/', '4', '5', '6', '*', '1', '2', '3', '-', '0', '.', '=', '+', 'C'].map((button) => (
          <button
            key={button}
            onClick={() => handleClick(button)}
            style={buttonStyle}
          >
```

```
                    {button}
                </button>
            ))}
        </div>
    </div>
  );
};

// Styles
const calculatorStyle = {
  width: '300px',
  margin: '50px auto',
  textAlign: 'center',
  fontFamily: 'Arial, sans-serif',
  border: '2px solid #ccc',
  borderRadius: '10px',
  padding: '20px',
  backgroundColor: '#f9f9f9',
};

const displayStyle = {
  marginBottom: '20px',
  textAlign: 'right',
  border: '1px solid #ccc',
  borderRadius: '5px',
  padding: '10px',
  backgroundColor: '#fff',
  fontSize: '18px',
  height: '60px',
  display: 'flex',
  flexDirection: 'column',
  justifyContent: 'space-between',
};

const inputStyle = {
  color: '#333',
};

const resultStyle = {
  color: '#007bff',
  fontSize: '22px',
  fontWeight: 'bold',
};

const buttonContainerStyle = {
  display: 'grid',
```

```
    gridTemplateColumns: 'repeat(4, 1fr)',
    gap: '10px',
};

const buttonStyle = {
    padding: '15px',
    fontSize: '18px',
    cursor: 'pointer',
    borderRadius: '5px',
    border: '1px solid #ddd',
    backgroundColor: '#007bff',
    color: 'white',
};

export default Calculator;
```

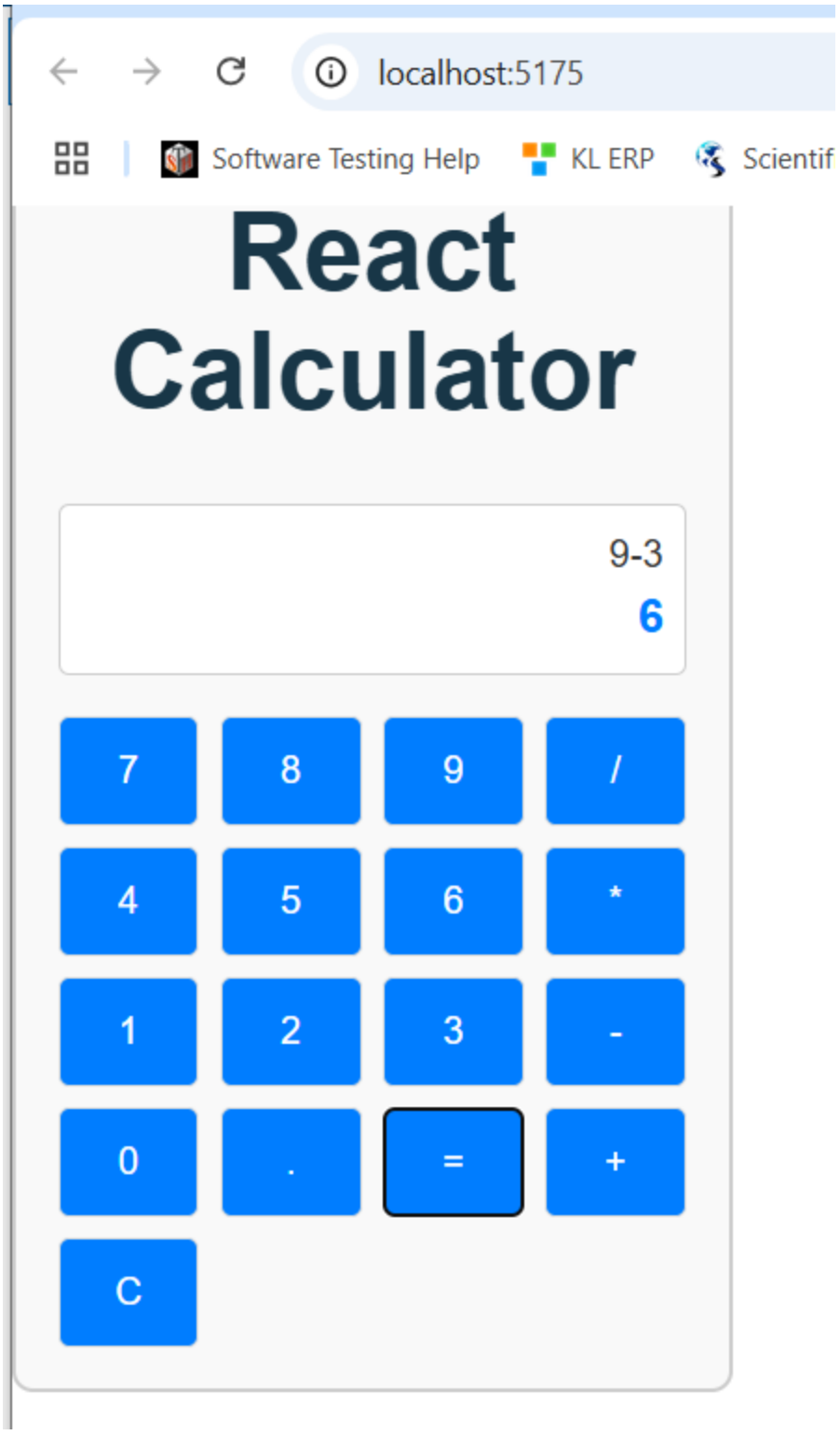
App.jsx

```
import React from 'react';
import Calculator from './Calculator';

const App = () => {
    return (
        <div>
            <Calculator />
        </div>
    );
};

export default App;
```

output :


localhost:5175
Software Testing Help
KL ERP
Scientif
React Calculator
9-3
6
7
8
9
/
4
5
6
*
1
2
3
-
0
.
=
+
C

Explanation:

Here is a detailed explanation of the React code for the calculator:

Imports

```
import React, { useState } from 'react';
```

1. **React**: The core library for building React components.
2. **useState**: A React Hook that allows us to add state management to functional components.

Component: Calculator

```
const Calculator = () => {
```

- **Calculator** is a functional component.
- This component handles the calculator's logic, state, and rendering.

State Management

```
const [input, setInput] = useState('');
const [result, setResult] = useState('');
```

1. **input**:
   - Stores the current input string (e.g., "12+3").
   - **Initial Value**: An empty string ('').
2. **setInput**:
   - Updates the input state whenever the user clicks a button.
3. **result**:
   - Stores the computed result of the expression.
4. **setResult**:
   - Updates the result state after evaluating the expression.

Button Click Handling

```
const handleClick = (value) => {
  if (value === '=') {
    try {
      const calcResult = eval(input);
      setResult(calcResult);
    } catch (error) {
      setResult('Error');
    }
  } else if (value === 'C') {
    setInput('');
    setResult('');
  } else {
    setInput(input + value);
  }
};
```

1. **handleClick**: Processes button clicks.
   - **value**: The button value clicked by the user.
2. **Actions**:
   - =:
     - Uses eval(input) to calculate the result.
     - If eval fails (e.g., invalid expression), the catch block sets the result to 'Error'.

        - **Note**: Avoid eval in production; use libraries like math.js for safety.
    - **C**:
        - Clears the input and result by resetting both states to empty strings.
    - **Other Values**:
        - Concatenates the clicked button's value to the input.

Rendering

Container Structure

```
<div style={calculatorStyle}>
  <h1>React Calculator</h1>
  <div style={displayStyle}>
    <div style={inputStyle}>{input}</div>
    <div style={resultStyle}>{result}</div>
  </div>
  <div style={buttonContainerStyle}>
    {['7', '8', '9', '/', '4', '5', '6', '*', '1', '2', '3', '-', '0', '.', '=', '+', 'C'].map((button) => (
      <button
        key={button}
        onClick={() => handleClick(button)}
        style={buttonStyle}
      >
        {button}
      </button>
    ))}
  </div>
</div>
```

1. **Calculator Container**:
    - Uses the calculatorStyle object for styling.
    - Contains the calculator's title, input/result display, and buttons.
2. **Display Section**:
    - **input**: Shows the user-typed mathematical expression.
    - **result**: Shows the computed result.
3. **Buttons**:
    - The map function iterates over the array of button values (digits, operators, special buttons).
    - **Props**:
        - **key**: A unique identifier for each button.
        - **onClick**: Attaches the handleClick function, passing the button value.
        - **style**: Applies button styles from buttonStyle.

Styling

1. **calculatorStyle**:
    - Centers the calculator on the screen and gives it a border, padding, and background color.
2. **displayStyle**:
    - A container for input and result, styled with borders and alignment.
3. **inputStyle**:
    - Styles the input text (current expression).
4. **resultStyle**:
    - Highlights the result text in blue with bold font.
5. **buttonContainerStyle**:
    - Uses CSS grid to arrange buttons in a 4x4 layout.
6. **buttonStyle**:

- Styles each button with padding, rounded corners, and a blue background.

Exporting

export default Calculator;

- Makes the Calculator component available for use in other files (e.g., App.js).

Flow

1. **Initial State**:
   - input and result are empty.
2. **Button Press**:
   - Updates input or computes the result.
3. **Display**:
   - Shows the current input and result dynamically.

Improvements

- Replace eval with a safer library like math.js.
- Add more operators (e.g., square root, exponentiation).
- Enhance the UI with animations or themes.

Let me know if you need further clarification!