React Hooks Explained

soumyabalamaala@gmail.com

Hooks are a feature introduced in React 16.8 that enable you to use state and other React functionalities in functional components, eliminating the need for class components.

They simplify the management of state and side effects in functional components.

soumyabalamaala@gmail.com

Rules of Hooks

Only Call Hooks at the Top Level:

Don’t call hooks inside loops, conditions, or nested functions. This ensures that hooks are called in the same order every time a component renders, which is crucial for React to correctly preserve the state of hooks between renders.

Only Call Hooks from React Functions:

You should only call hooks from React functional components or custom hooks. This means you can’t call hooks from regular JavaScript functions or class components.

soumyabalamaala@gmail.com

List of Hooks

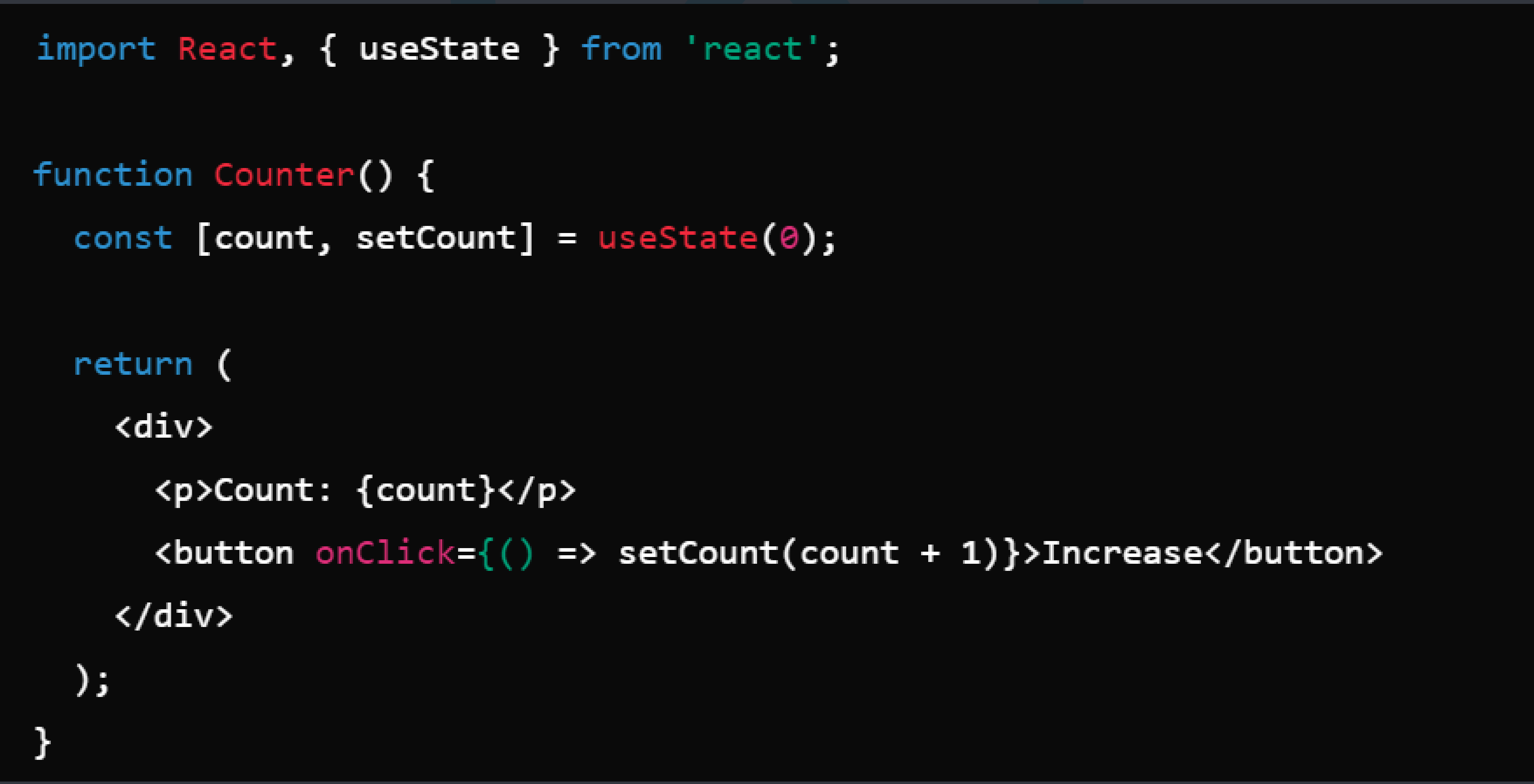
1. useState ()
2. useEffect ()
3. useContext ()
4. useMemo ()
5. useRef ()
6. useReducer ()

soumyabalamaala@gmail.com

useState ()

Purpose:

Manages state within a functional component.



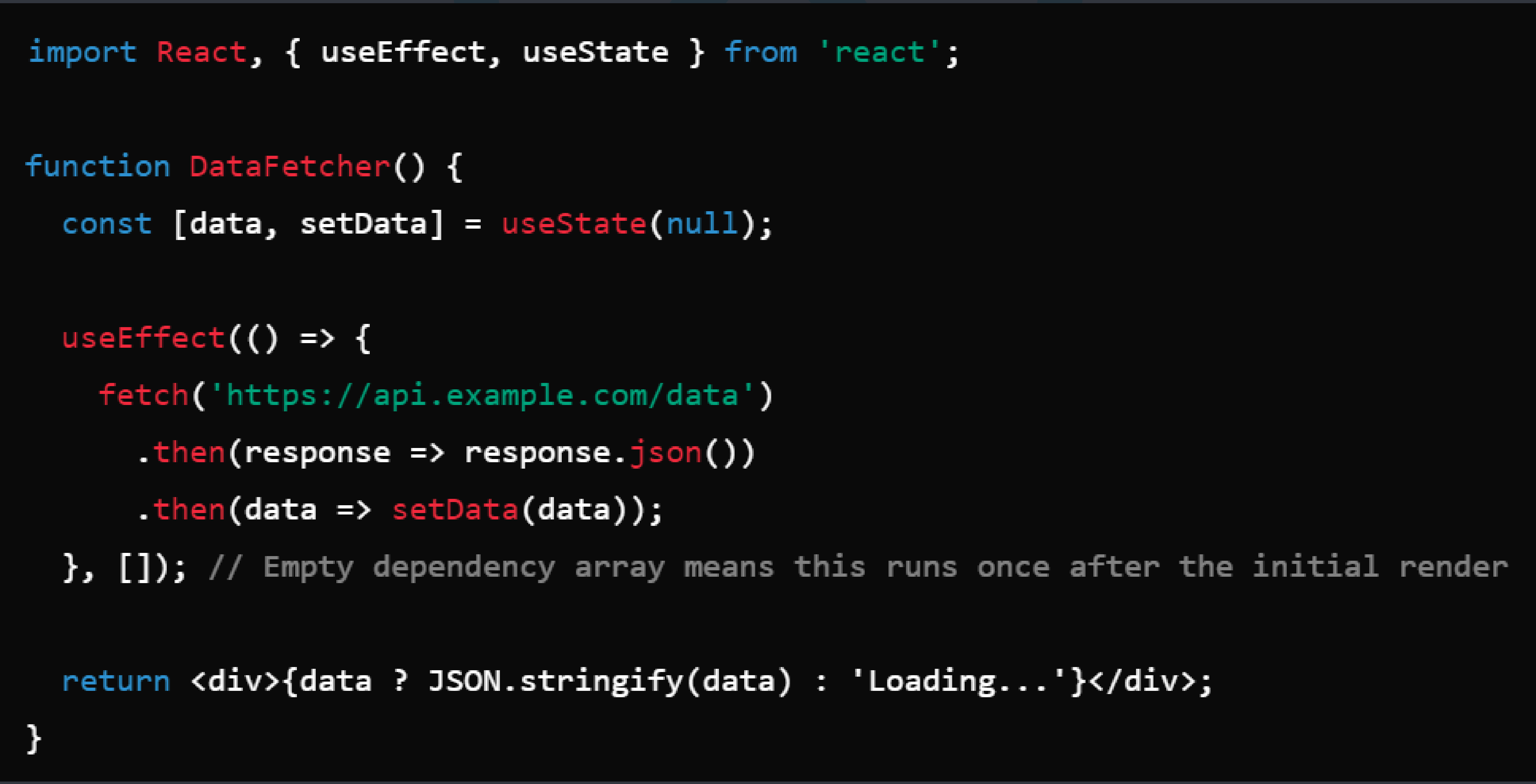
```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increase</button>
    </div>
  );
}
```

soumyabalamaala@gmail.com

useEffect ()

Purpose:

Performs side effects like data fetching, subscriptions, or manually manipulating the DOM.



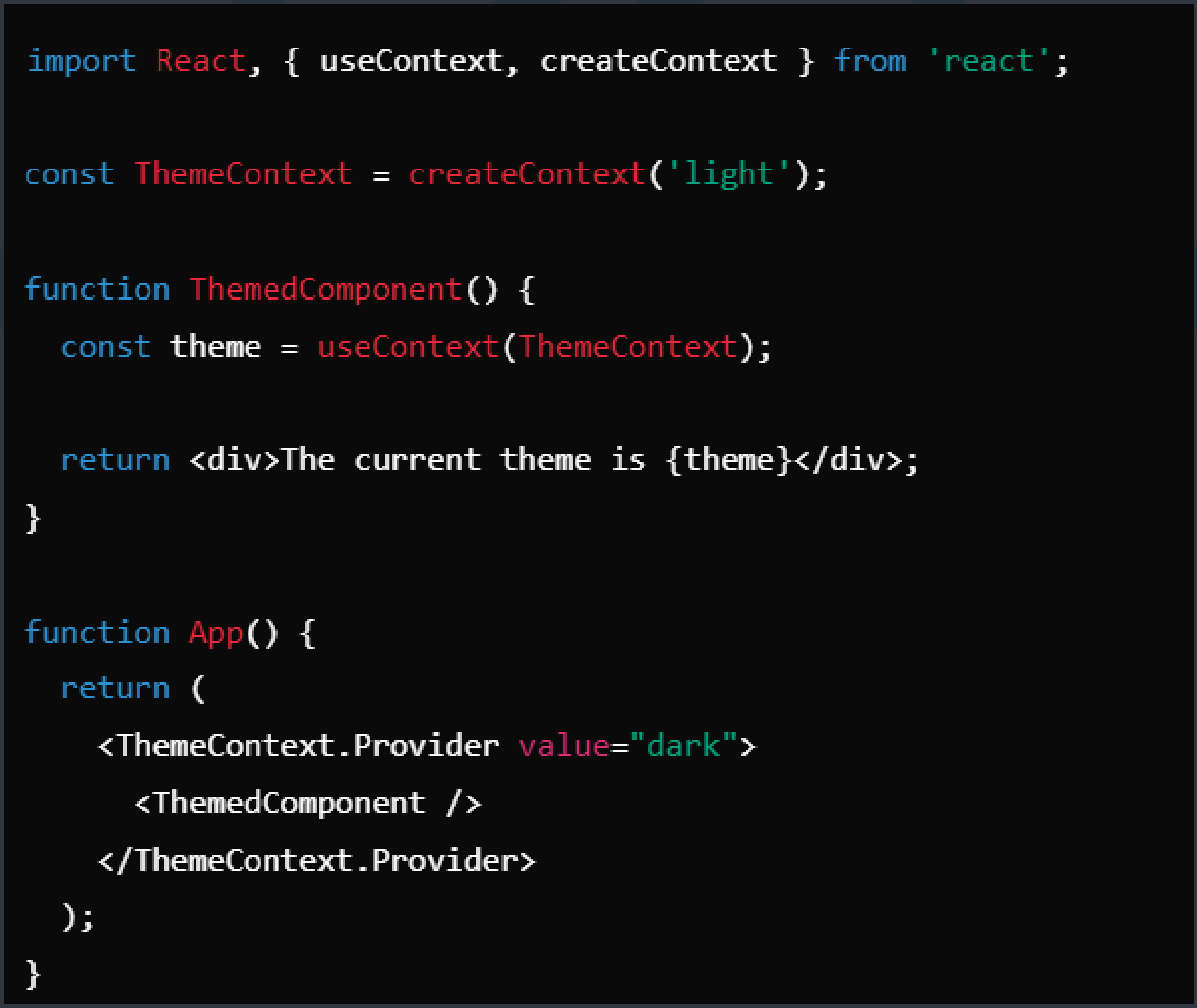
```
import React, { useEffect, useState } from 'react';

function DataFetcher() {
  const [data, setData] = useState(null);

  useEffect(() => {
    fetch('https://api.example.com/data')
      .then(response => response.json())
      .then(data => setData(data));
  }, []); // Empty dependency array means this runs once after the initial render

  return <div>{data ? JSON.stringify(data) : 'Loading...'}</div>;
}
```

soumyabalamaala@gmail.com

useContext ()

Purpose:

Accesses the value of a context created with **React.createContext()**



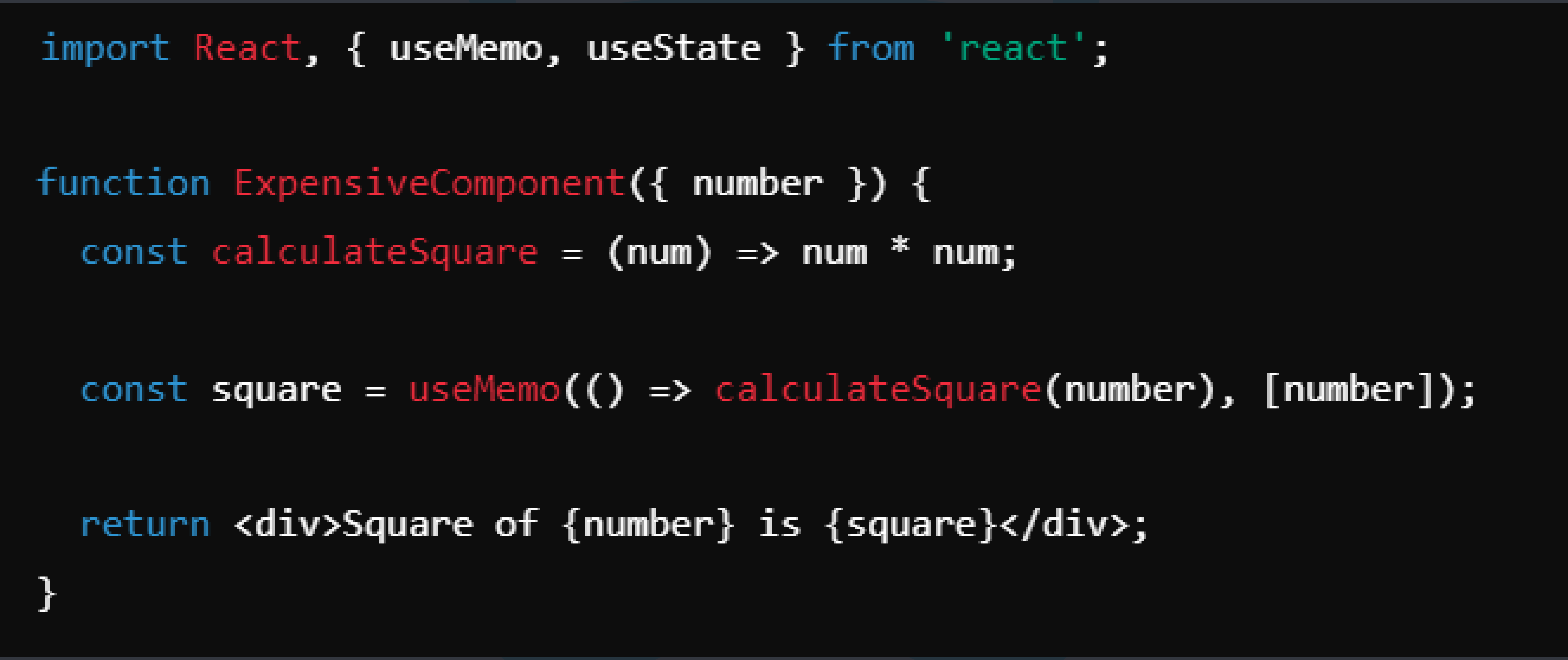
```
import React, { useContext, createContext } from 'react';

const ThemeContext = createContext('light');

function ThemedComponent() {
  const theme = useContext(ThemeContext);

  return <div>The current theme is {theme}</div>;
}

function App() {
  return (
    <ThemeContext.Provider value="dark">
      <ThemedComponent />
    </ThemeContext.Provider>
  );
}
```

soumyabalamaala@gmail.com

useMemo ()

Purpose:

Memoizes a value to optimize performance by recalculating it only when dependencies change.



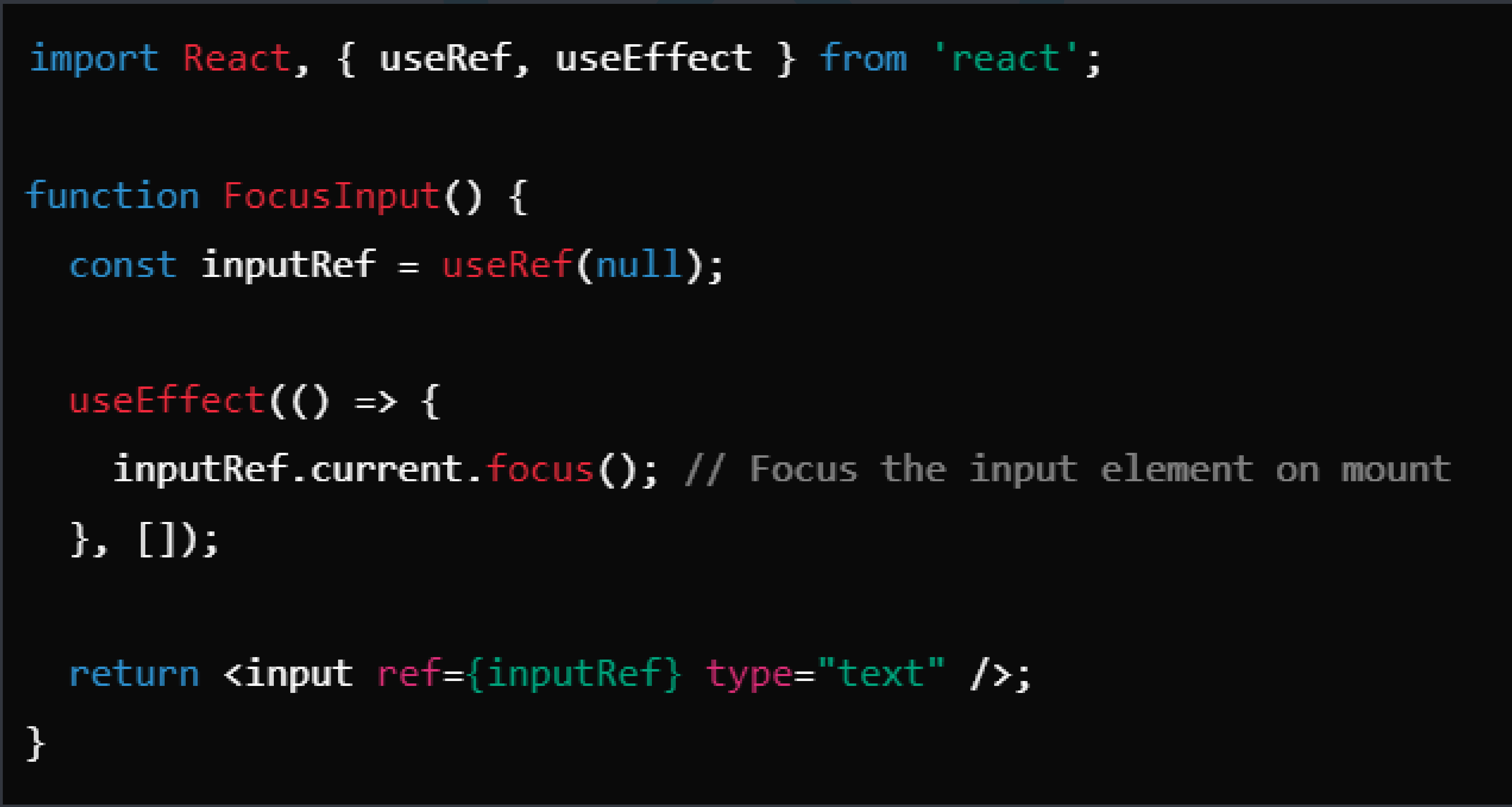
```
import React, { useMemo, useState } from 'react';

function ExpensiveComponent({ number }) {
  const calculateSquare = (num) => num * num;

  const square = useMemo(() => calculateSquare(number), [number]);

  return <div>Square of {number} is {square}</div>;
}
```

soumyabalamaala@gmail.com

useRef ()

Purpose:

Provides a mutable ref object that persists across renders and can be used to access DOM nodes or store mutable values.



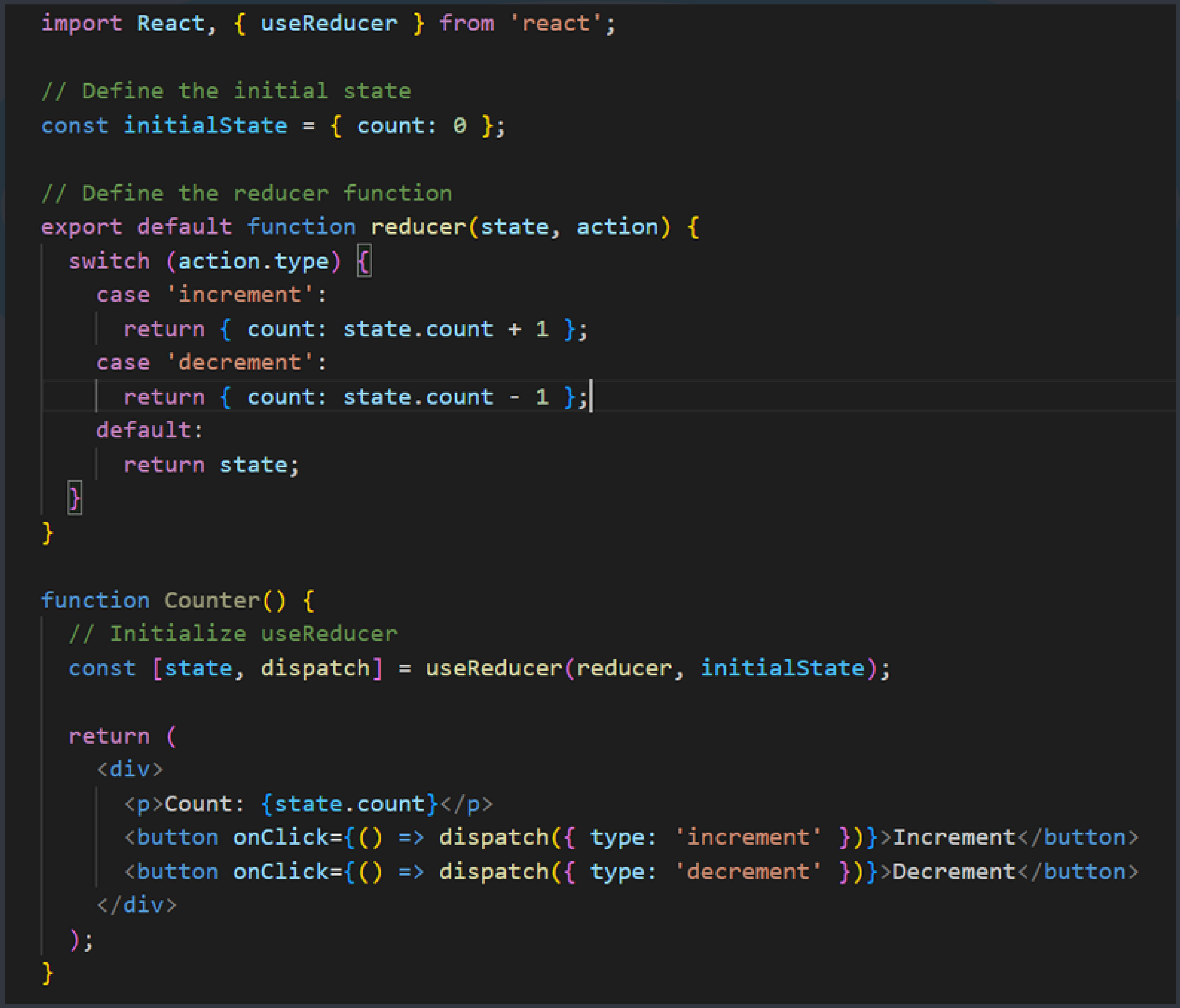
```
import React, { useRef, useEffect } from 'react';

function FocusInput() {
  const inputRef = useRef(null);

  useEffect(() => {
    inputRef.current.focus(); // Focus the input element on mount
  }, []);

  return <input ref={inputRef} type="text" />;
}
```

soumyabalamaala@gmail.com

useReducer ()

Purpose:

Manages state with a reducer function, which provides a more structured approach to state updates compared to useState.



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

// Define the initial state
const initialState = { count: 0 };

// Define the reducer function
export default function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      return state;
  }
}

function Counter() {
  // Initialize useReducer
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'increment' })}>Increment</button>
      <button onClick={() => dispatch({ type: 'decrement' })}>Decrement</button>
    </div>
  );
}
```

soumyabalamaala@gmail.com

Have a
Project in Mind ?

soumyabalamaala@gmail.com