|  |
| --- |
| import React, { useState } from "react";  const ExpensiveCalculation = () => {  const [count, setCount] = useState(0);  const [number, setNumber] = useState(1000000);  const calculateSum = () => {  console.log("Calculating...");  let sum = 0;  for (let i = 0; i < number; i++) {  sum += i;  }  return sum;  };  const sum = calculateSum();  return (  <div>  <h1>Sum: {sum}</h1>  <button onClick={() => setCount(count + 1)}>Increment Count: {count}</button>  </div>  );  };  export default ExpensiveCalculation; |
| import React, { useState, useMemo } from "react";  const ExpensiveCalculationOptimized = () => {  const [count, setCount] = useState(0);  const [number, setNumber] = useState(1000000);  const sum = useMemo(() => {  console.log("Calculating...");  let sum = 0;  for (let i = 0; i < number; i++) {  sum += i;  }  return sum;  }, [number]);  return (  <div>  <h1>Sum: {sum}</h1>  <button onClick={() => setCount(count + 1)}>Increment Count: {count}</button>  </div>  );  };  export default ExpensiveCalculationOptimized; |
| useMemo(() => expensiveFunction(), [dependency1, dependency2]); |
| ****Difference Between**** useMemo ****and**** useCallback  | **Feature** | **useMemo** | **useCallback** | | --- | --- | --- | | **Purpose** | Memoizes the **result** of a computation. | Memoizes a **function** definition. | | **Return Value** | Returns the computed value. | Returns the memoized function itself. | | **Use Case** | Avoids recalculating expensive computations. | Prevents unnecessary function re-creations. | | **Dependency Array** | Recomputes the value only if dependencies change. | Recreates the function only if dependencies change. | |
| import React, { useState } from "react";  const Child = React.memo(({ onClick }) => {  console.log("Child rendered");  return <button onClick={onClick}>Click Me</button>;  });  const Parent = () => {  const [count, setCount] = useState(0);  const handleClick = () => {  console.log("Button clicked");  };  return (  <div>  <h1>Parent Count: {count}</h1>  <button onClick={() => setCount(count + 1)}>Increment Parent Count</button>  <Child onClick={handleClick} />  </div>  );  };  export default Parent; |
| import React, { useState, useCallback } from "react";  const Child = React.memo(({ onClick }) => {  console.log("Child rendered");  return <button onClick={onClick}>Click Me</button>;  });  const Parent = () => {  const [count, setCount] = useState(0);  const handleClick = useCallback(() => {  console.log("Button clicked");  }, []); // Dependency array ensures function is recreated only if dependencies change  return (  <div>  <h1>Parent Count: {count}</h1>  <button onClick={() => setCount(count + 1)}>Increment Parent Count</button>  <Child onClick={handleClick} />  </div>  );  };  export default Parent; |
|  |