In React, `useMemo` and `useCallback` are Hooks that help optimize the performance by memoizing values. They both accept a function as an argument and return a memoized version of the function.

The main difference is while `useMemo` returns a memoized value that is the result of running the function, `useCallback` returns a memoized callback function that only changes if one of the dependencies has changed.

useMemo

React `useMemo` is a hook that lets you cache the result of a calculation between re-renders. It is useful when you want to avoid expensive calculations on every render, by caching the previous result.

```
const cachedValue = useMemo(calculateValue, dependencies)
```

```
import { memo } from 'react';


const SomeComponent = memo(function SomeComponent(props) {

// ...

});
```

useCallback

React `useCallback` is a hook that lets you cache a function definition between re-renders. This can improve performance by preventing unnecessary re-rendering of components that depend on the function.

```
const cachedFn = useCallback(fn, dependencies)
```