

# React's 6 Main Hooks Explained

Next →

# useState

Create and update state values.

holds the  
state value

used to update  
the state value

```
const [count, setCount] = useState(0)  
setCount(1)
```

changes the  
count value to 1

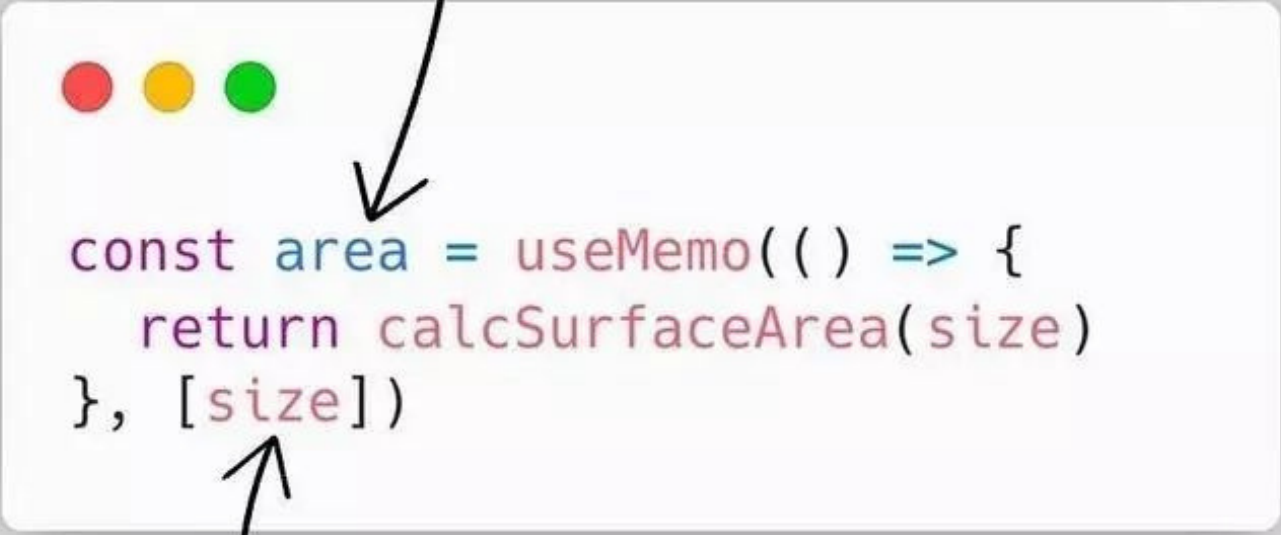
creates the  
state resources  
and sets the  
initial value to 0

Next →

# useMemo

Returns a memoized value which only gets recalculated when the defined dependencies change.

holds the cached value returned by calcSurfaceArea



```
const area = useMemo(() => {  
  return calcSurfaceArea(size)  
}, [size])
```

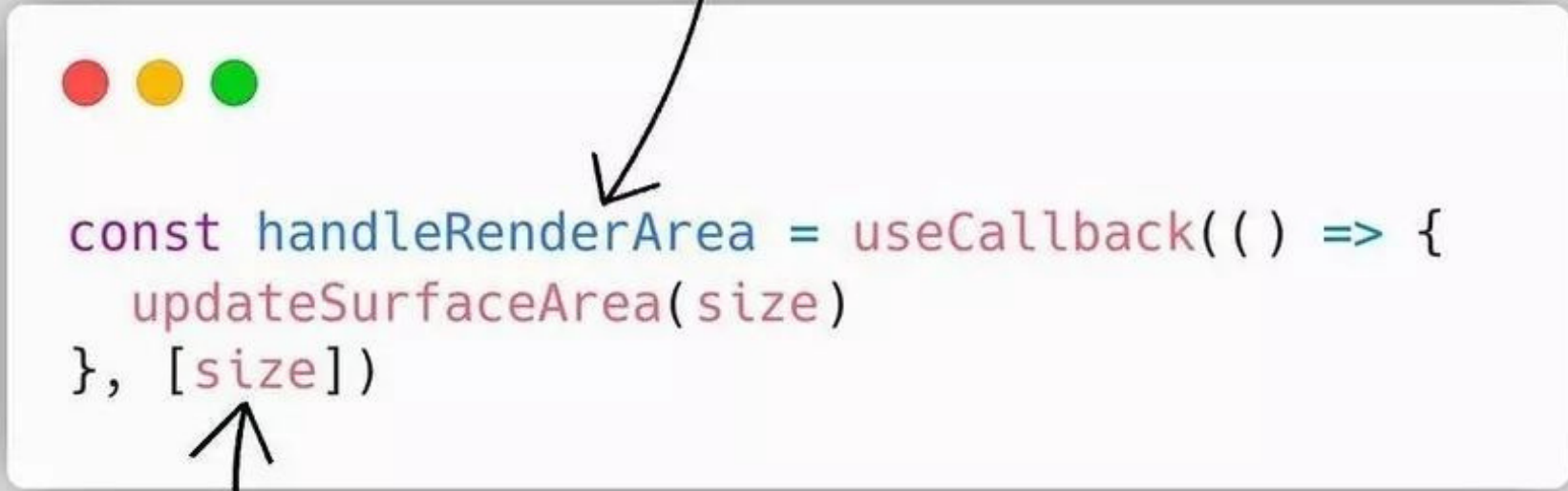
area updates every time size changes



# useCallback

Returns a memoized version of a callback that only changes when the dependencies change.

a memoized version of  
updateSurfaceArea

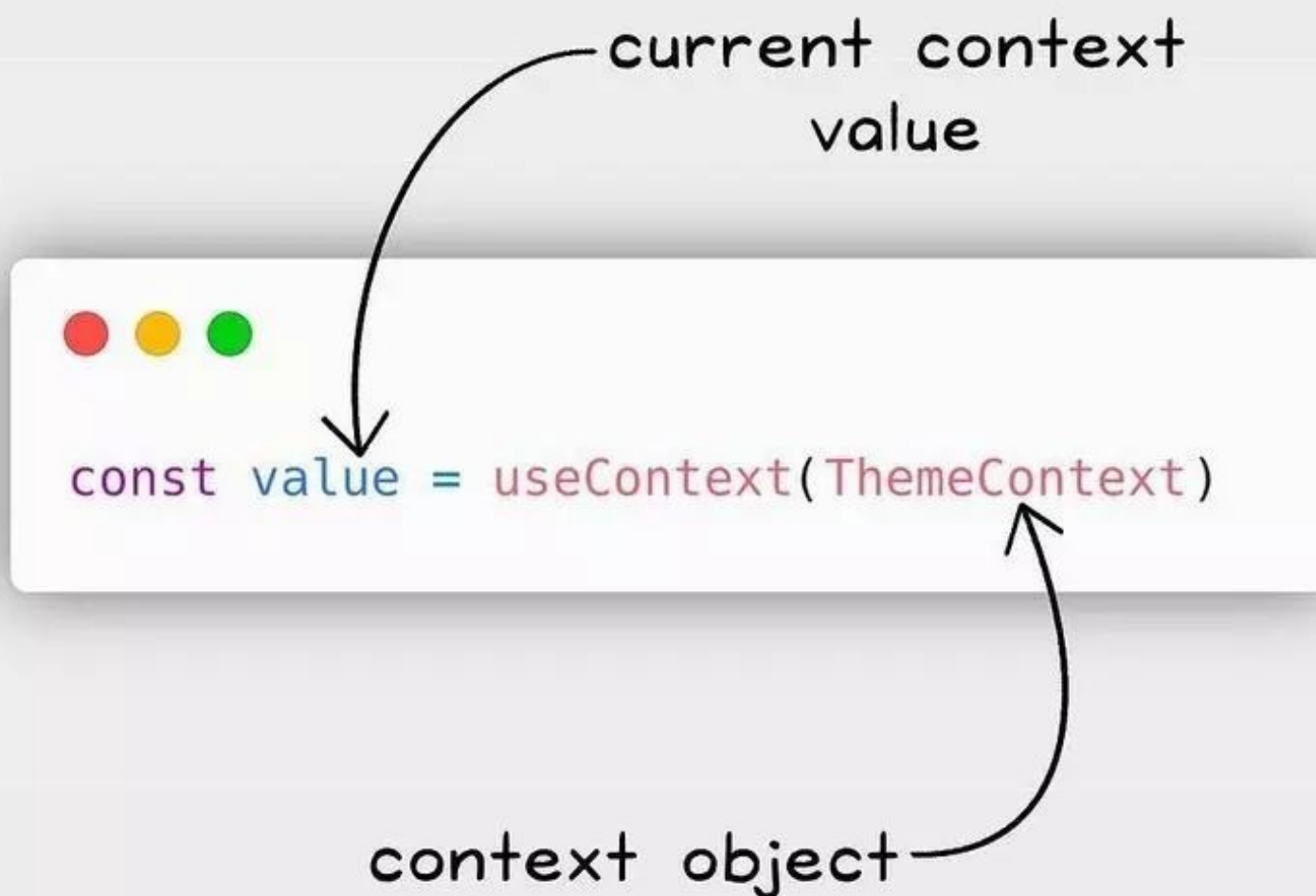


```
const handleRenderArea = useCallback(() => {  
  updateSurfaceArea(size)  
}, [size])
```

handleRenderArea  
updates when size  
changes value

# useContext

Accepts a context object that's created using `React.createContext`, and returns the current value of that context.



# useEffect

Used to run side effects in the component such as fetching data or adding listeners.

runs after the initial render

runs just before the component unmounts

```
useEffect(() => {  
  → addListeners()  
  return () => {  
    removeListeners()  
  }  
})  
  
useEffect(() => {  
  → fetchUserInfo(userID)  
}, [userID])
```

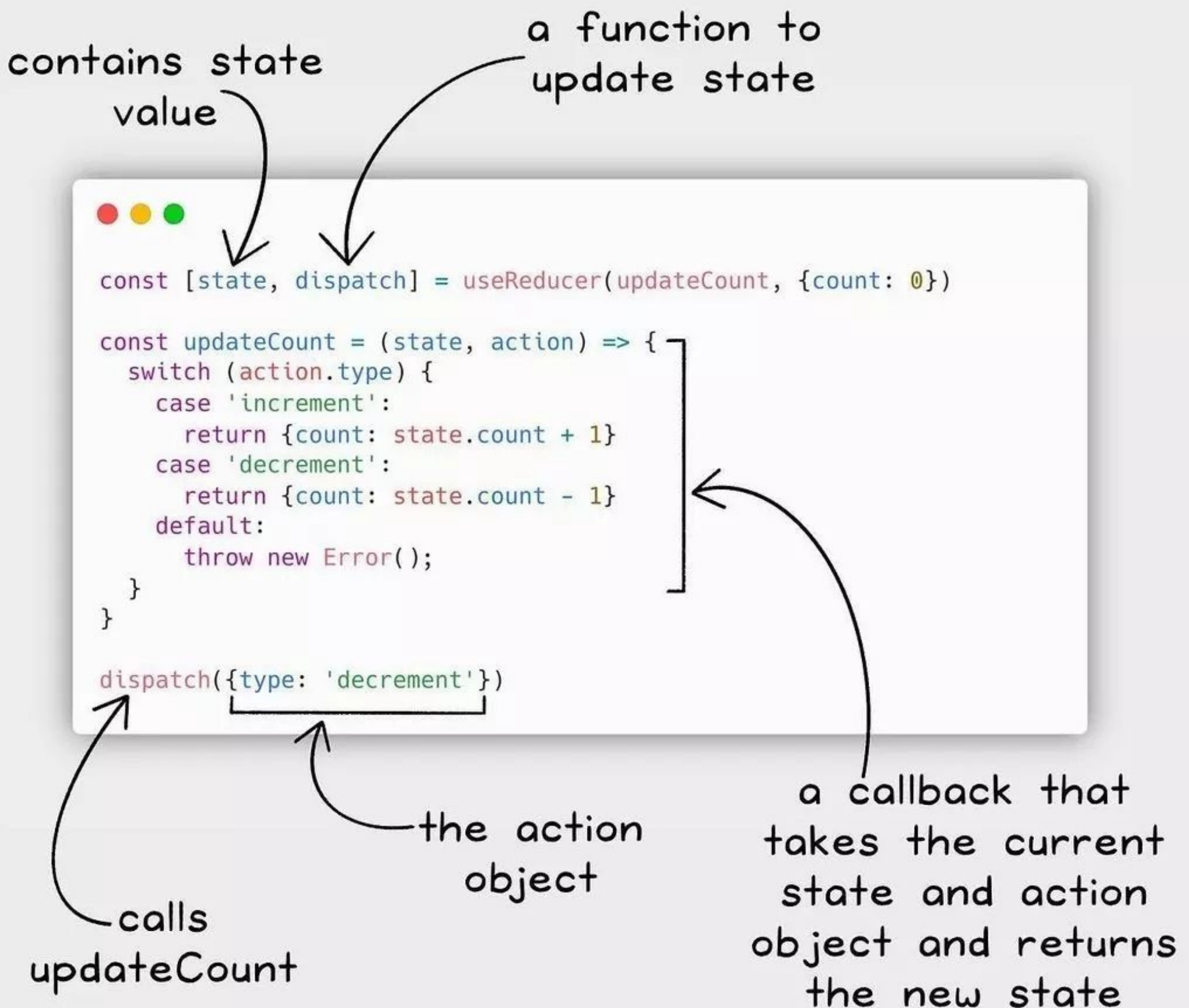
runs after the first render and every time userID updates

Next →



# useReducer

Similar to useState but also let's you use your own update state logic.





# Abhishek Mankuskar



Frontend Web & App Developer || Social Winter Of Code (SWOC) Founding  
Organiser || DevRel 🥑 || Open Source Contributor & Community Builder |

**Follow for more such content.**

**Please Let me know what do  
you think about this post.**



Next →