Hooks	useState	useEffect	useLayoutEffect	useContext	useCallback	useMemo	useReducer	useRef

Syntax	const [stateN, setStateN] = useState(initalState)	useEffect(()=>{
		// This can be a set up function.
		// can be a cleanup function
		},[dependency array])
			useLayoutEffect(()=>{
			},[dependency array])
				Const value=useContext(context)	Const cacheFunc=useCallback(fn, dependencies)	Similar to the useCallback function but returns a value instead of a function.	useReducer(<reducer><initialState>)	const ref= useRef(initialValue)

Syntax Explanation	1. setStateN does not return any value.
	2. initialState can be a pure function but should not take an argument and must have a return type.
	3. setStateN is an async function, so updates are not directly visible immediately after its usage. It is possible via updater function.
	Ex.
	If n = 5
	setStateN(n=>n-1) //4
	setStateN(n=>n-1)//3
		1. No dependency array –
		useEffect(()=>{
		
		})
		Means the callback function iside the useffect will be executed in every React state cycle.
		2. No element in dependency array –
		useEffect(()=>{
		},[])
		This means the callback function will be executed only while mounting phase.
		3. Non-empty dependency array.
		useEffect(()=>{
		},[a,b])
		This means if either of a or b changes the callback function will be called at mounting as well as updating phase.
		4. First code for clean up gets executed then the setup function.
				1. Import useContext from 'react'
				
				Const useContext=createContext()
				
				In the file where you want to you use the useContext.
				2. We must wrap the tree component with context provider.
				
				Function App(){
				const[user, setUser] = useState('test')
				return{
				<UserContext.Provider value = {user}>
				<Comp2 user={user}>
				</UserContext.Provider>
				}
				}
				3. Other components can use –
							reducer – contains custom state logic.
							
							initialState – simple value but generally contains an object.
							
							Returns current state and dispatch method
								Syntax is not straightforward
								Import {useRef} from 'react'
								
								Let ref = useRef(0)
								
								Const click=()=>{
								Ref.current = ref.current + 1
								
								}
								
								Return{
								
								<button onClick={click}> Click </button>
								}-

Why?	useState hook is used for – 1. Updating the in-built React state object value. 2. Adding a state to a component.	useEffect is used when you want to connect your component with API or an external system. Or when you want side effects like removing a UI element based on certain condition.	It is fired only after DOM loading is complete. useLayoutEffect() works only at the time of component mounting and update. Generally used to select UI elements and directly working on them.	To solve prop drilling issue. To use React-redux easily. Easily used with useState hook and can be passed to any nested component directly.	Returns a memorized callback function. Isolate resource intensive functions so that they will not run on every re-render. useCallback runs when one of the dependencies changes	It is used to keep expensive function from running needlessly	For better state management. Especially for a complex state management.	To manipulate Dom. To stop recreating ref contents between the renders
Limitation	One should be mindful of using the useState setter function i.e. denoted by setStateN() only inside a function component. If the new set value is similar to the old value, then React discards re-rendering.	The useEffect dependency array must only contain state variables. They are less efficient than middleware and can also cause race condition. Having no dependency array can cause infinite re-rendering issue.						

Example	1. Updating object using React set function –
	const[obj, setObj] = useState({}) --declaration
	
	setObj({...obj, {name:"test", age: "100"}}) // setting object value using spread operator.
	
	2. Updating array using React set function –
	
	const[name,setName] = useState([]) // declaration
	
	setName([...name, "test"]) //setting array elements using spread operator
	why we need spread operator?
	
	React state is read-only. Spread operator create a copy of existing object and all the added values are added sequentially.							