React

useCallback

Hook

Few points to consider before we get started with useCallback Hook

➡ In JavaScript arrays, objects & functions are of reference types (Not primitive types like string, number or boolean)

➡ So, even if you have two similar objects / arrays / functions it's not identical in JavaScript

➡ Example: [1, 2] === [1, 2]
// Outputs: false

Consider a scenario where there's a Parent component and it has only one Child component

➡ What if the only prop that the Child component receives is a callback function which remains the same during re-renders

➡ You may think to wrap the Child component with React.memo to save it from future re-renders (If the props has not changed)

➡ But, as I have mentioned earlier that in case of functions (Although they are identical) React memo would fail

➡ Because on the next re-render cycle a fresh copy of function is created which gets passed to the Child component. React memo will treat that as new prop and hence it will not stop the Child component from re-rendering

Syntax

➡ useCallback is a hook which returns the memoized version of the callback function which only changes when one of it's dependancy changes (Kind of similar to useEffect Hook)

```
const memoizedCB = useCallback ( () => {...}, [] )
```

Important Note ⬇⬇

useCallback Hook is useful when we want pass callback Functions as props to the optimized Child components (with React.Memo) to prevent them From unnecessary re-renders

Do you find this post helpful then please do share this post with your connections ;))