BROSWER OPT.

→ Website optimization is a process of improving a website's loading speed in the browser.

What is Memoization?

- **Memoization** is a speed optimization technique in programming, where given a function, you return a cached version of the output if the same inputs are used.
- A memoized function **remembers** the results of an output for a given set of inputs.
- In **React**, we can memoize components (where the inputs are props), functions, or just a regular computed value.
- When memoizing components, it does a **shallow** comparison (by default) of the props, and if it sees that it has changed, then it will re-render.
- Memoization is not free. You are trading space for time.

→ There are basically three reasons over react comp. is re-render

→ state is changed
→ props is changed
→ if any changed occure in react parent's comp that trigger re-render to child comp

React.memo()

→ React memo is higher-order comp. that wraps around a component to memoize the rendered output and avoid unnecessary rendering. This improves performance because it memoizes the result and skips rendering to reuse the last render result.

There are two ways you can wrap your component with React.memo

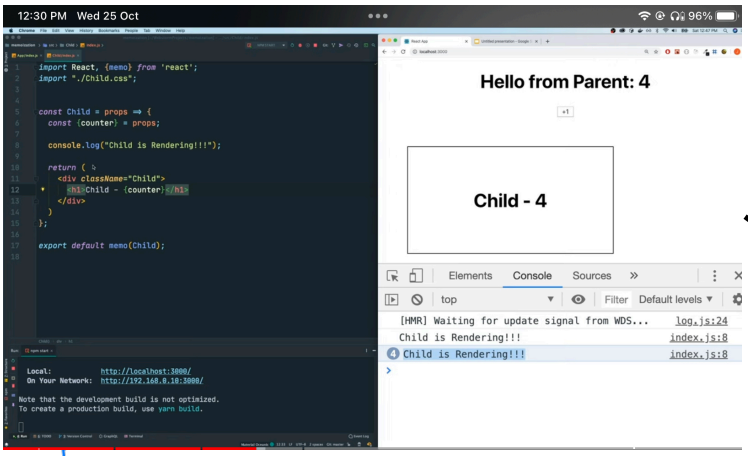
```
const myComponent = React.memo((props) => {
    /* render using props */
});

export default myComponent;
```

Another option is to create a new variable to store the memoized component and then export the new variable:

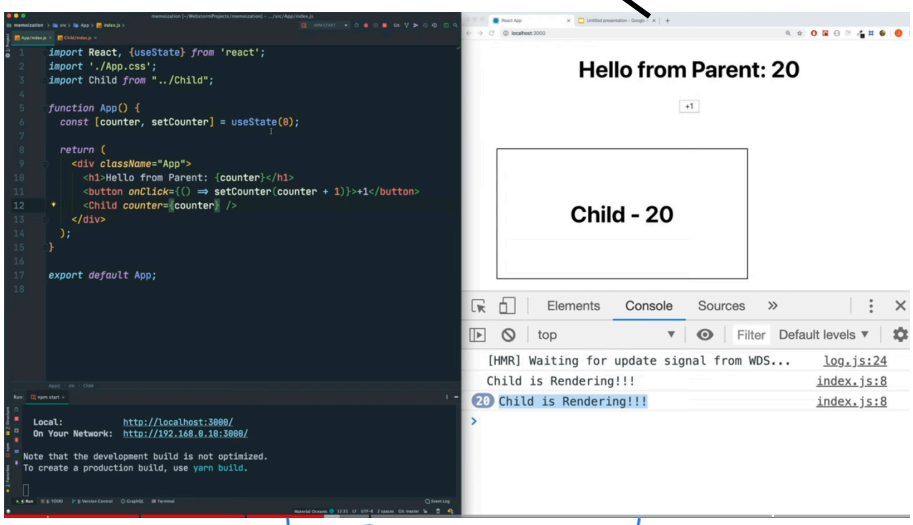
```
const myComponent = (props) => {
    /* render using props */
};

export const MemoizedComponent = React.memo(myCom
```

This is the example of React.memo()

React.memo() Example

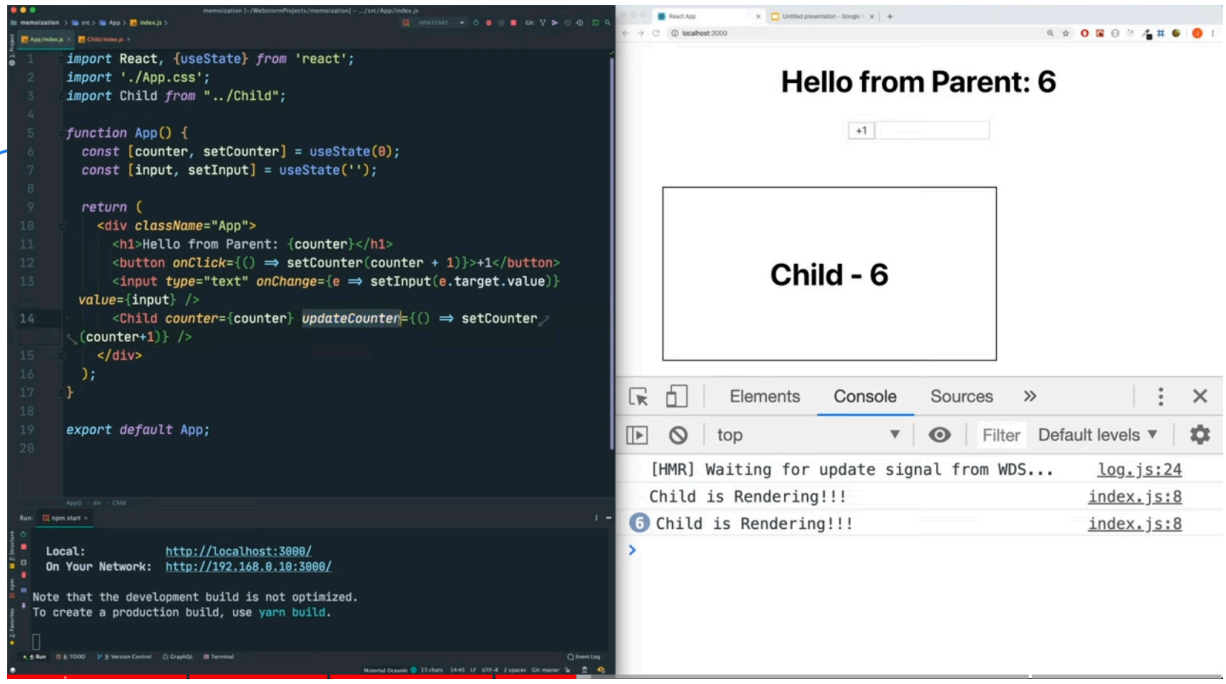


Child



Parents

2)



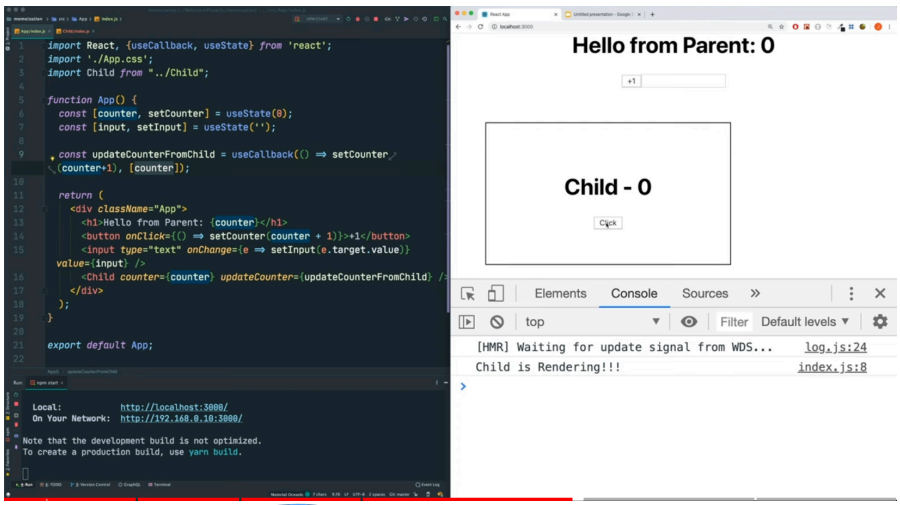
→ When we giving any value inside input field on that time child is rendering because it is find the new reference of function.... So, here you have to memorize the function in React.

For memorizing the function we have to use useCallback().

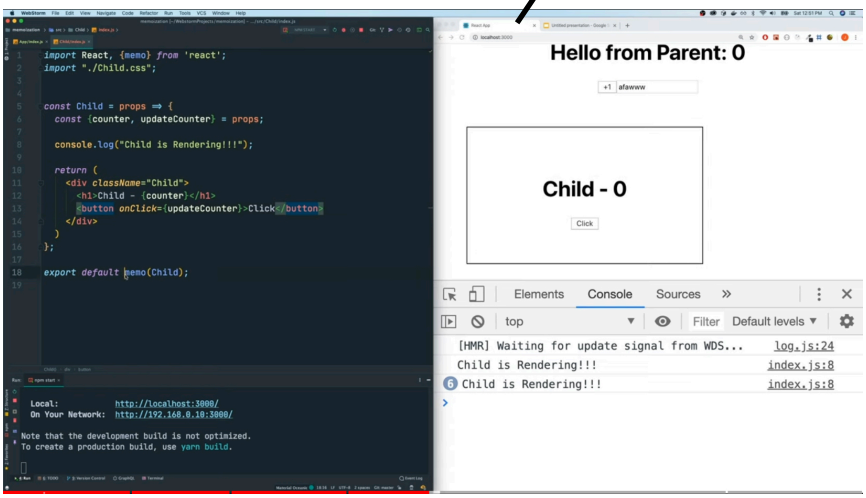
This hook is used to optimize a React application by returning a memoized function which helps to prevent unnecessary re-rendering of a function. This hook stores the cached value of the function and only updates the function if the passed dependencies changes.

Syntax

```
const memoizedCallback = useCallback(() => {doSomething(a, b); }, [a, b],);
```



Parent



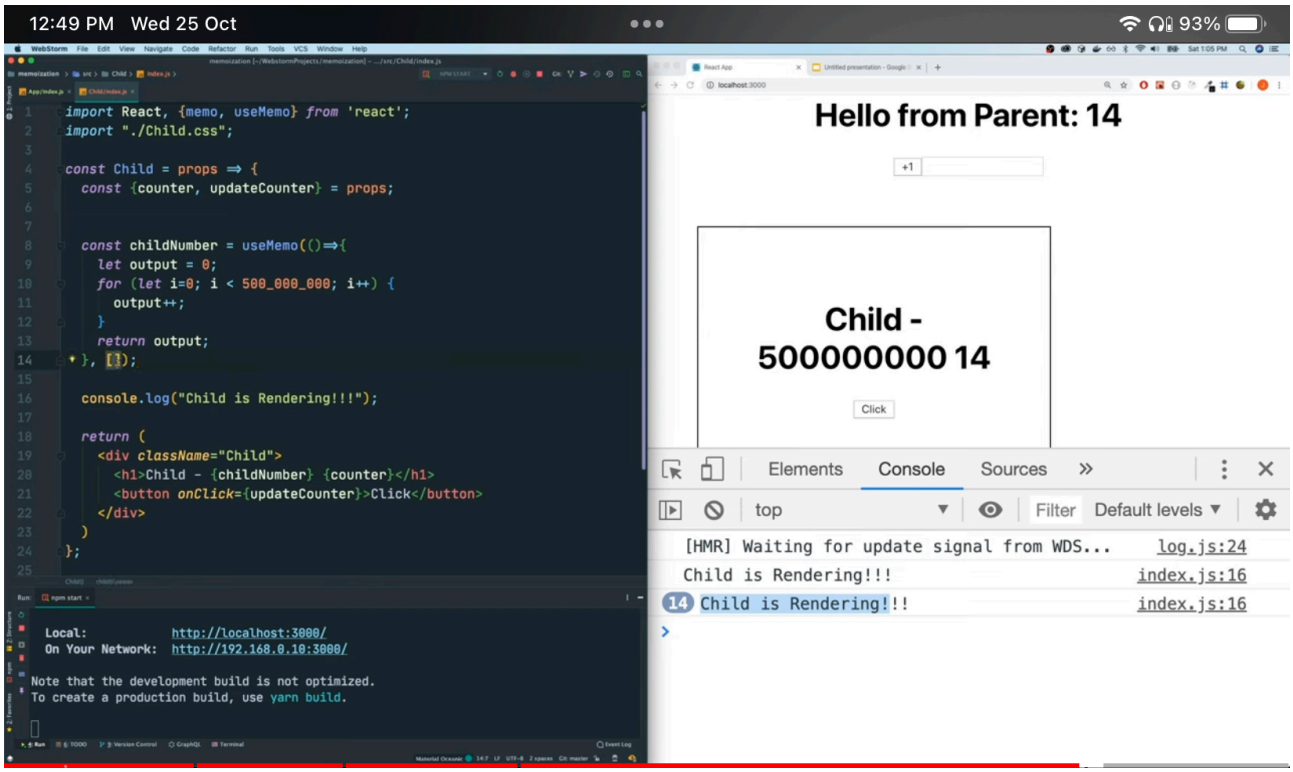
Child

useMemo()

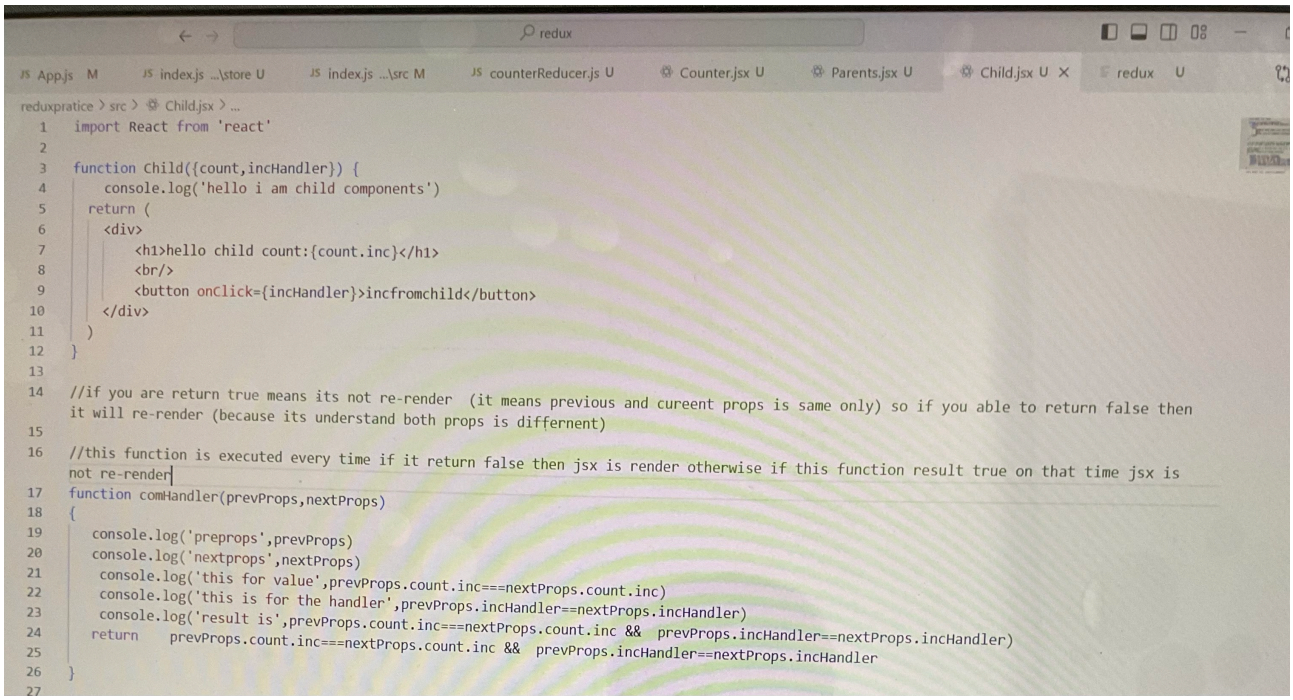
useMemo() Hook

`useMemo` will only recompute the memoized value when one of the dependencies has changed. This optimization helps to avoid expensive calculations on every render.

→ useMemo is same like useCallback()



More details about second argument in React.memo()



→ If function return true that time re-render is not happening, if it false then re-render is happening