React memo, useCallback and useMemo

React.Memo : 1) React memo checks for changes between the previous and current values for a given prop

passed to the component.  
 2) it prevent un-necessary render when child component is rendering with parent

component.   
 Ex :  
import React, { useCallback, useState } from 'react'

import Child from './Child'

const Parent = () => {

    const [count, setCount] = useState(0)

    let name = 'satyam';

    const increment = () => setCount(count + 1)

    const fun = () => console.log('Hi Guys!! I am from calling from function ')

    return (

        <div>Parent: {count}

            <Child />

            <button onClick={increment}>increment</button>

        </div>

    )

}

export default Parent

import React, { memo } from 'react'

const Child = ({name}) => {

    console.log('from child component');

    return (

        <div>Child component Hi: {name}</div>

    )

}

// here I used memo for prevent the un-necessary render

export default memo(Child)

The main difference is that useMemo returns a memoized value and useCallback returns a memoized function.

useCallback:

1. **useCallback** is used to memoize functions so that they are not recreated on every render unless their dependencies change.
2. It is particularly useful when passing callbacks to child components it prevent unnecessary re-renders of those components.

Ex :

import React, { useCallback, useState } from 'react'

import Child from './Child'

const Parent = () => {

    const [count, setCount] = useState(0)

    let name = 'satyam';

    const increment = () => setCount(count + 1)

    // for this fun function when i was passing in child component then

    // again every click on increment calling child component in inspect

    // for overcome this challenge we have to useCallback

    // const fun = () => console.log('Hi Guys!! I am from calling from function ');

    // \*\*\*\*\*\*\*\*\*\*\*\* Before calling to useCallback\*\*\*\*\*\*\*\*\*\*\*

    // when we using useCallback then un-neccessary render is not happen

    const fun = useCallback(() => {

        console.log('Hi Guys!! I am from calling from function ')

    }, [])

    return (

        <div>Parent: {count}

            <Child name={name} onfun={fun} />

            <button onClick={increment}>increment</button>

        </div>

    )

}

export default Parent

import React, { memo } from 'react'

const Child = ({name, onfun}) => {

    console.log('from child component');

    return (

        <div>Child component Hi: {name}</div>

    )

}

export default memo(Child)

**useMemo:**useMemo returns a memoized value

1) This hook is used to memoize the result of a function

2) It returns a memoized value that only changes if one of the dependencies has changed.

3) It avoid expensive calculations on every render.

Ex:   
import React, { useState, useMemo } from 'react';

function ExpensiveCalculation({ count }) {

// Use useMemo to memoize the result of the (function) expensive calculation

const result = useMemo(() => {

console.log('Performing expensive calculation...');

return count \* 2;

}, [count]); // Recompute the result only if 'count' changes

return <p>Result of Expensive Calculation: {result}</p>;

}

function ParentComponent() {

const [count, setCount] = useState(0);

return (

<div>

<button onClick={() => setCount(prevCount => prevCount + 1)}>Increment Count</button>

<ExpensiveCalculation count={count} />

<p>Parent Count: {count}</p>

</div>

);

}

export default ParentComponent;