**Memo**

What is memo ?

Memo is a react function that is use to render a component when it should be otherwise the component will not render

What is render ?

When we do some action then the whole code is taken from the top to bottom. Then it is called render.

Why we use memo?

Suppose there is a child component in the app.js and if we don’t want to render the child component because of the rendering of the app.js then we use memo in the child component and that stops the child to render because of the render of app.js

How we use memo?

In the app.js

import React, { useState } from 'react'

import Child from './component/child';

import child from './component/child'

function App() {

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

console.log("app rendering")

  return (

    <div>

        <h1>Count is : {count}</h1>

        <button onClick={()=>{

          setcount(count+1)

        }}>increase</button>

        <Child }/>

    </div>

  )

}

export default App

and in the child .js

import React from 'react'

function child() {

    console.log("msg rendering")

  return (

    <div>

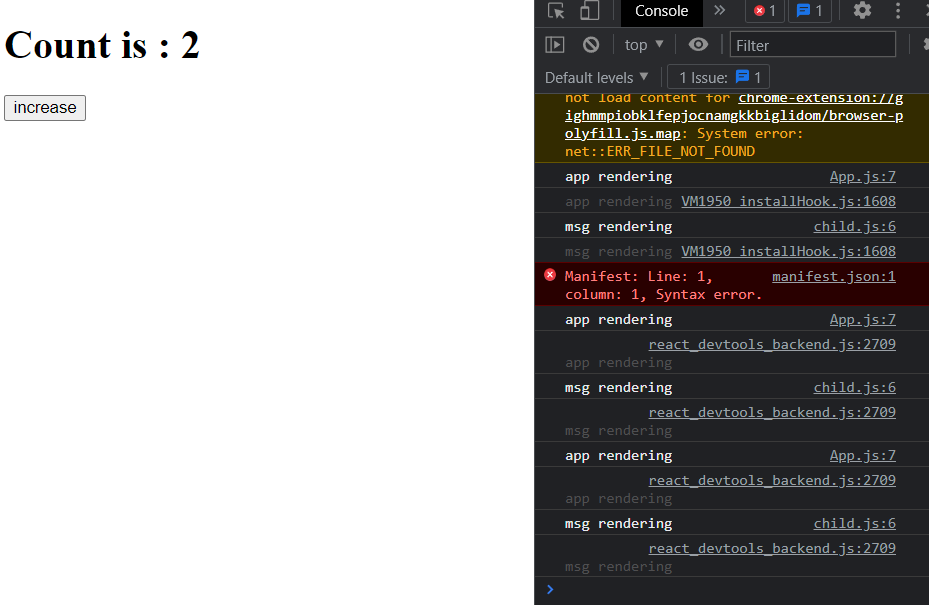
    </div>

  )

}

export default child

OUTPUT:



As we see in the output when we click the increase button then the child and parent will also rendering that not we want

That’s why we use memo in the child section

import React, { memo } from 'react'

function child() {

    console.log("msg rendering")

  return (

    <div>

Use memo in the child section function

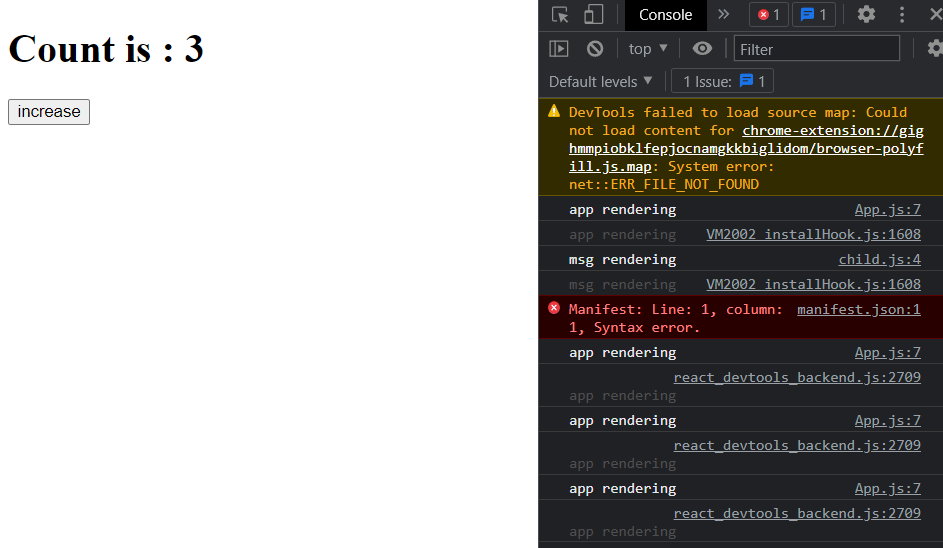
    </div>

  )

}

export default memo(child)

after using memo in the child section



When we click the increase button then only app is rendering but msg rendering is not shown

**useCallback**

in app.js

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

import Child from './component/child'

function App() {

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

When we use this callback function then the count will not render by the cause of other variable like toggle.

When we don’t use useCallback function then the changing of toggle can render the child and that’s what we don’t want and that’s why we use this useCallback function

const [toggle, settoggle] = useState(false);

let handleIncreaseCount=useCallback(()=>{

  setcount(count+1)

},[count])

console.log("app rendering")

  return (

    <div>

        {

          toggle? "ON" : "OFF"

        }

        <button onClick={()=>{

          settoggle(!toggle)

        }}>toggle</button>

        <h1>Count is : {count}</h1>

        <button onClick={()=>{

          setcount(count+1)

        }}>increase</button>

        <Child msg={count} increaseCount={handleIncreaseCount}/>

    </div>

  )

}

export default App

in child.js

import React, { memo } from 'react'

function child({msg,increaseCount}) {

    console.log("msg rendering")

  return (

    <div>

        <p>message is {msg}</p>

        <button onClick={()=>{

            increaseCount()

        }} >increase from child</button>

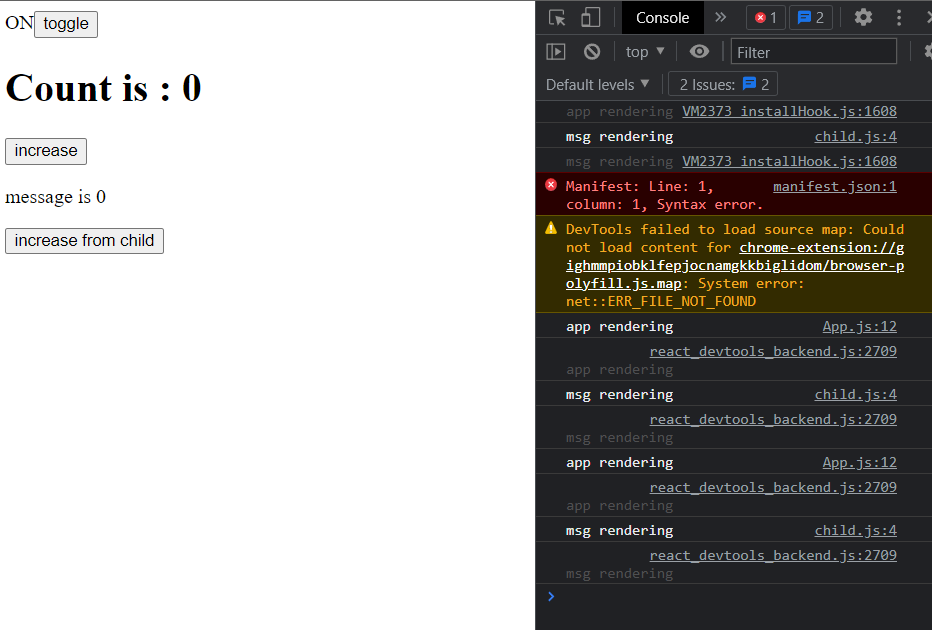
    </div>

  )

}

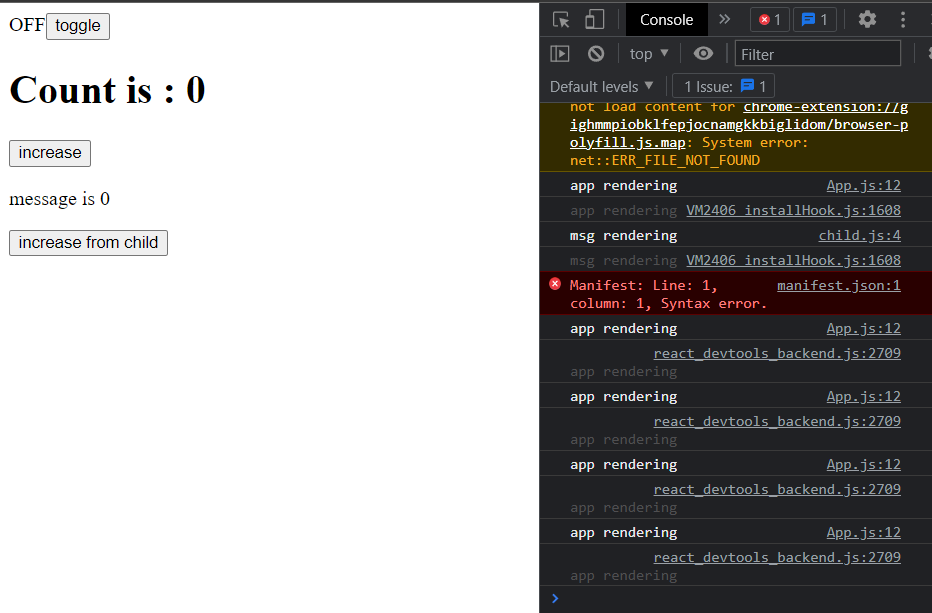
export default memo(child)

before using useCallback function:



Changing of toggle cause both app rendering and msg rendering

After using useCallback function



Changing of toggle cause only app rendering

**useMemo**

why we use useMemo?

= suppose if we want to calculate a value for only one time and after that we don’t need to calculate it when it is rendering or the app.js is rendering then we use for a specific section in the useMemo hook

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

import Child from './component/child'

function App() {

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

  const [toggle, settoggle] = useState(false);

  // useCallback function

  let handleIncreaseCount = useCallback(() => {

    setcount(count + 1)

  }, [count])

Here, we use useMemo to calculate a number for once and it don’t calculate the same number when the app.js is rendering.

Here the calculatedNumber calculate the whole number and when the app.js is rendering the calculatedNumber don’t calculate the same number

  // useMemo

  let calculatedNumber = useMemo(() => {

    let number = 0;

    for (let index = 0; index < 500000000; index++) {

      number++;

    }

    return number;

  }, [])

  console.log("app rendering")

  return (

    <div>

      {

        toggle ? "ON" : "OFF"

      }

      <button onClick={() => {

        settoggle(!toggle)

      }}>toggle</button>

      <h2>NUmber is : {calculatedNumber}</h2>

      <h1>Count is : {count}</h1>

      <button onClick={() => {

        setcount(count + 1)

      }}>increase</button>

      <Child msg={count} increaseCount={handleIncreaseCount} />

    </div>

  )

}

export default App