**Lesson 01 Demo 03**

**useMemo hook example**

**Objective:** To demonstrate the useMemo hook to optimize the performance

**Tools required:** Node JS and React JS

**Prerequisites:** HTML, CSS,JavaScript ES5/ES6, Basic React Concept

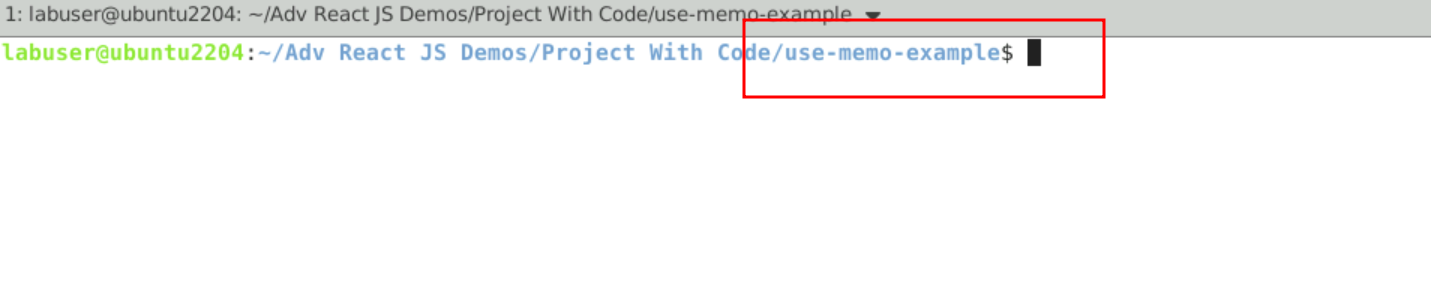
**Note** : All react js project already created with version 18.x with Sample App.js file

**Steps to be followed:**

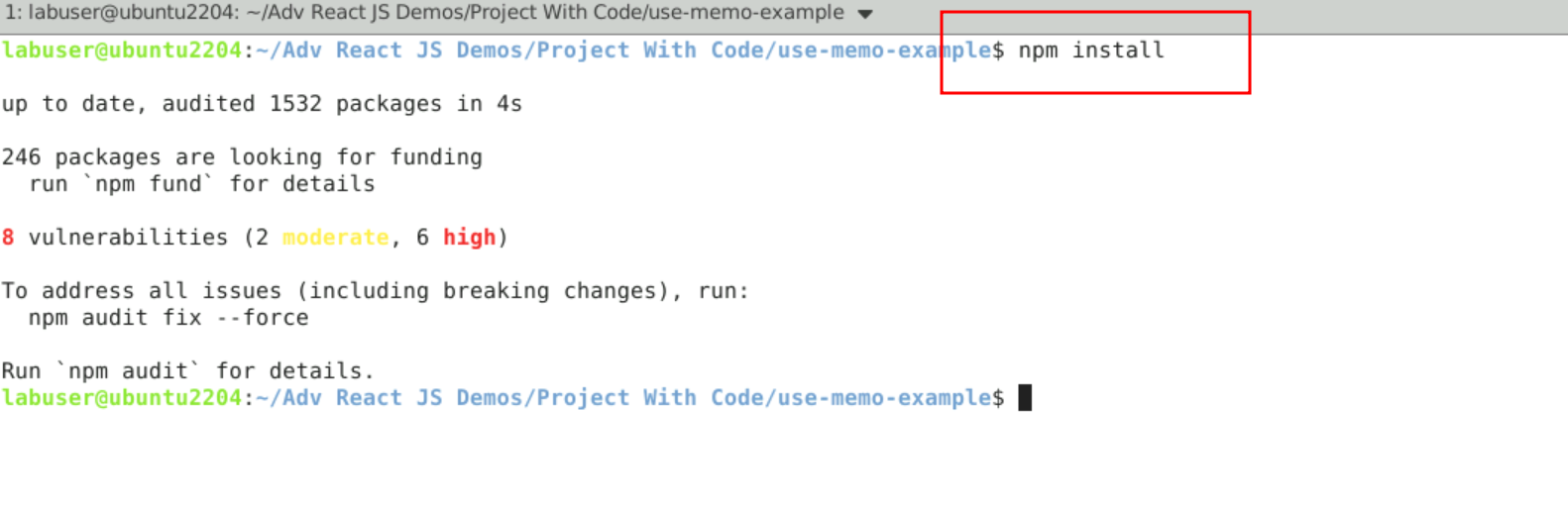
1. Set up for react js project
2. What is the use of useMemo hook.
3. Doing expensive function call without useMemo
4. Doing expensive function call with useMemo hook
5. Doing expensive function call with useMemo with dependencies.
6. Adding All three components file in App.js
7. Testing App.js file without useMemo expensive function call.
8. Testing App.js file with useMemo expensive function call.
9. Testing App.js file with useMemo expensive function call with dependencies

**Step 1: Set up for react js project**

1. Open a terminal window inside a React JS pre-created project ie **use-memo-example**

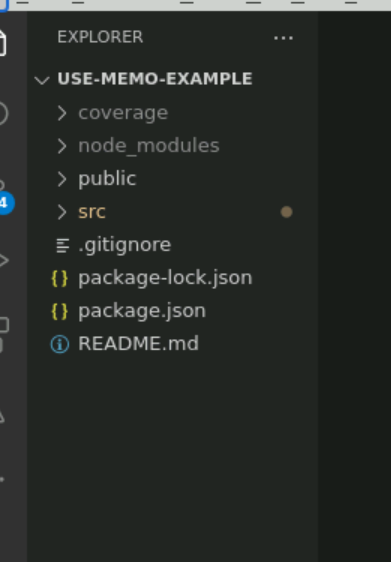


1. Now you need to run the command as **npm install.** This command helps us to installed all required dependencies mention in package.json file in local machine in the form of node\_module folder.



1. Now open **use-memo-example** folder in VS Code Editor

Note: short cut to open write **code .**



**Step 2: What is the use of useMemo hook**.

1. The useMemo hook in React is used to optimize the performance of your components by memoizing the results of expensive calculations. It allows you to cache the result of a function and only recompute it when the dependencies of the function have changed.

2.2. The ‘useMemo’ hook takes two arguments: a function and a array of dependecies. the function is the one that performs the expensive computation, and the dependencies determine when the memoized value should be recomputed.

**Step 3: Doing expensive function call without useMemo**

3.1 This component is responsible to execute the loop 10000 time and adding the value in data values. This function call again and again.

3.2 Below code you need to write inside a **WithoutUseMemo.js** file

import React, { useState } from 'react'

import './App.css'

function WithoutUseMemo() {

console.log('re-render')

const [someBool,setSomeBool] = useState(false)

const expensiveFunction=()=>{

console.log('run expensive function')

const data=[]

for(let i=0;i<=10000;i++){

if(i%2===0){

console.log(i)

data.push(i)

}

}

return data

}

// Run This First and See The OutPut in Console

const calculationData=expensiveFunction()

return (

<div>

<div>current Value:{someBool?'true':'false'}</div>

<button onClick={()=>setSomeBool(prevState=>!prevState)}>change</button>

<hr/>

{calculationData&&calculationData.map((el)=>{

return <p key={el}>{el}</p>

})}

</div>

)

}

export default WithoutUseMemo;

**Step 4: Doing expensive function call with useMemo hook**

4.1 This component is responsible to execute the loop 10000 time and adding the value in data values. This function call again and again. This function calling with the help of useMemo.

4.2 Below code you need to write inside a **WithUseMemo.js** file

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

import './App.css'

function WithUseMemo() {

console.log('re-render')

const [someBool,setSomeBool] = useState(false)

const expensiveFunction=()=>{

console.log('run expensive function')

const data=[]

for(let i=0;i<=10000;i++){

if(i%2===0){

console.log(i)

data.push(i)

}

}

return data

}

// Run This First and See The OutPut in Console

// const calculationData=expensiveFunction()

// After This Execute This With useMemo and See the Difference

const calculationData = useMemo(()=>expensiveFunction(),[])

return (

<div>

<div>current Value:{someBool?'true':'false'}</div>

<button onClick={()=>setSomeBool(prevState=>!prevState)}>change</button>

<hr/>

{calculationData&&calculationData.map((el)=>{

return <p key={el}>{el}</p>

})}

</div>

)

}

export default WithUseMemo;

**Step 5: Doing expensive function call with useMemo hook with dependencies**

5.1 This component is responsible to execute the loop 10000 time and adding the value in data values. This function call again and again. This function calling with the help of useMemo with state dependencies variable.

5.2 Below code you need to write inside a **UseMemoWithDependency.js** file

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

import './App.css'

function UseMemoWithDependency() {

console.log('re-render')

const [someBool,setSomeBool] = useState(false)

const [amount,setAmount] = useState(1000)

const expensiveFunction=()=>{

console.log('run expensive function')

const data=[]

for(let i=0;i<=amount;i++){

if(i%2===0){

console.log(i)

data.push(i)

}

}

return data

}

// Run This First and See The OutPut in Console

// const calculationData=expensiveFunction()

// After This Execute This With useMemo and See the Difference

const calculationData = useMemo(()=>expensiveFunction(),[amount])

return (

<div>

<div>current Value:{someBool?'true':'false'}</div>

<button onClick={()=>setSomeBool(prevState=>!prevState)}>change bool</button>

{/\* Lets Add Some data Now \*/}

<button onClick={()=>setAmount(prevState=>prevState+100)}>Add Data</button>

<hr/>

{calculationData&&calculationData.map((el)=>{

return <p key={el}>{el}</p>

})}

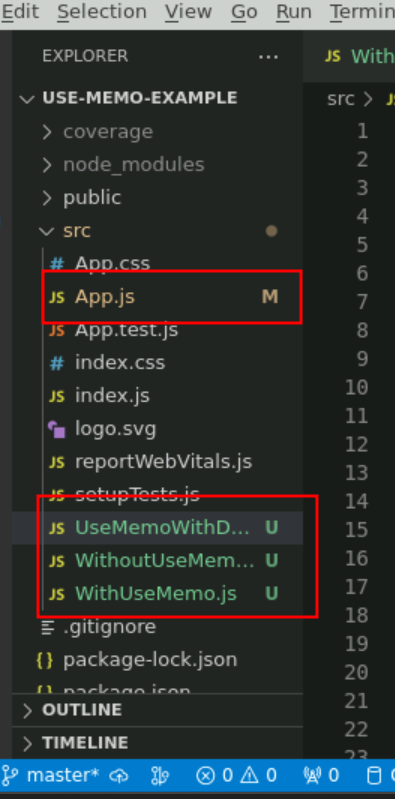
</div>

)

}

export default UseMemoWithDependency;

5.3 After created all files you can see the project structure.



**Step 6 : Adding All three components file in App.js**

6.1 open the app.js file.

import logo from './logo.svg';

import './App.css';

import WithoutUseMemo from './WithoutUseMemo';

import WithUseMemo from './WithUseMemo';

import UseMemoWithDependency from './UseMemoWithDependency';

function App() {

  return (

    <div className="App">

      <WithoutUseMemo></WithoutUseMemo>

      <WithUseMemo></WithUseMemo>

      <UseMemoWithDependency></UseMemoWithDependency>

    </div>

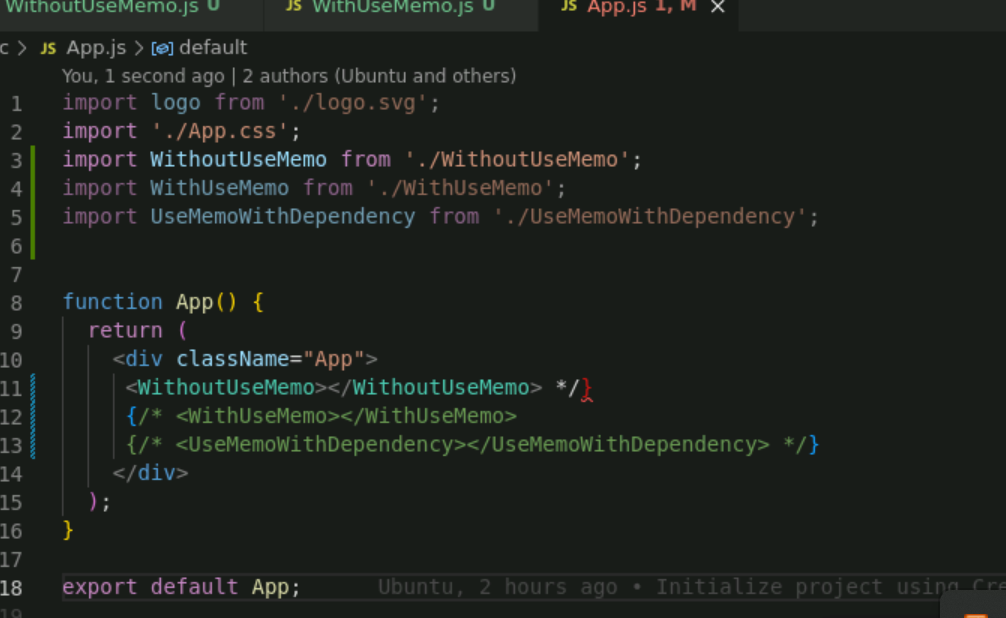
  );

}

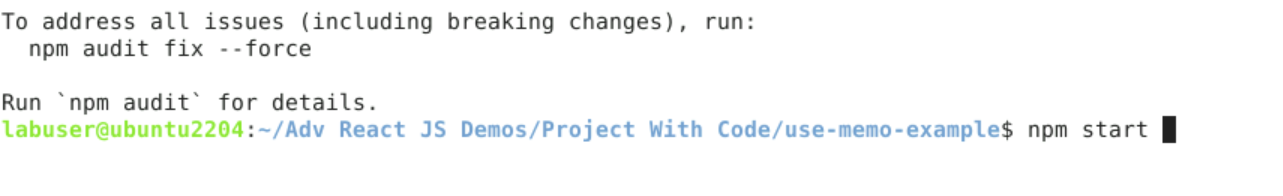
export default App;

**Step 7. Testing App.js file without useMemo expensive function call.**

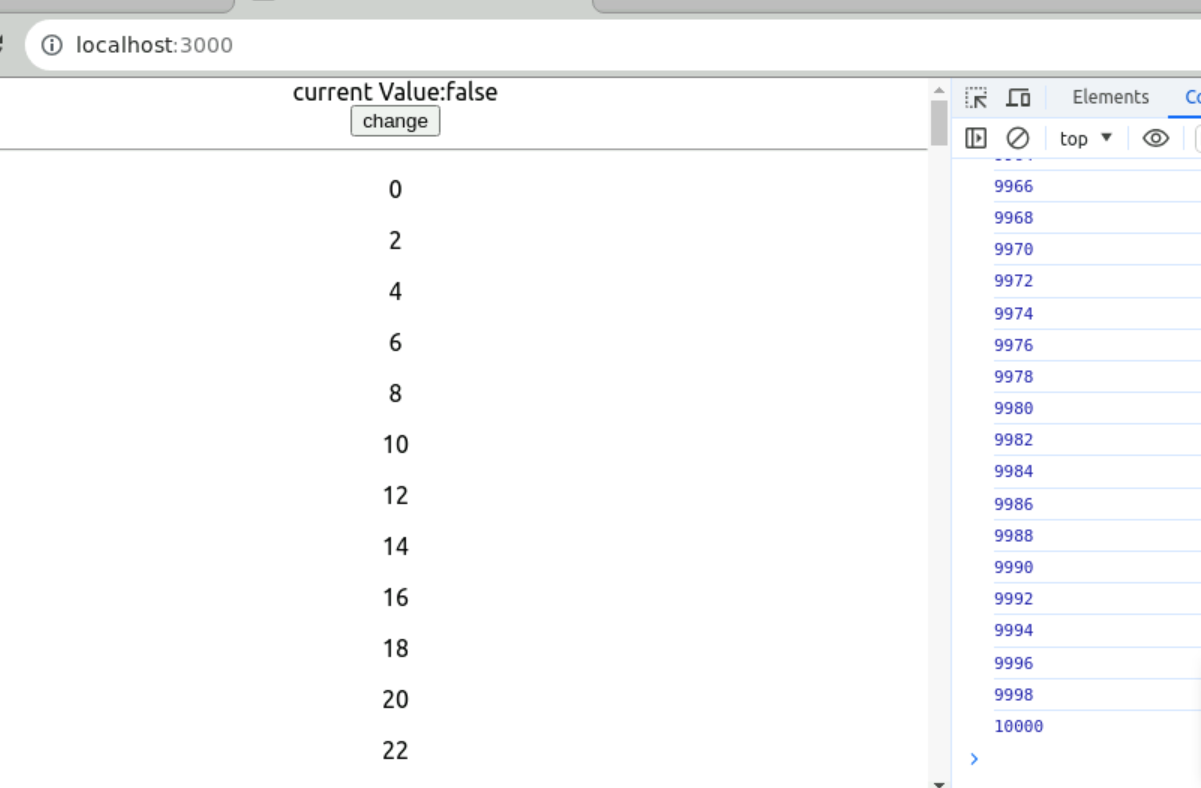
7.1 App.js file



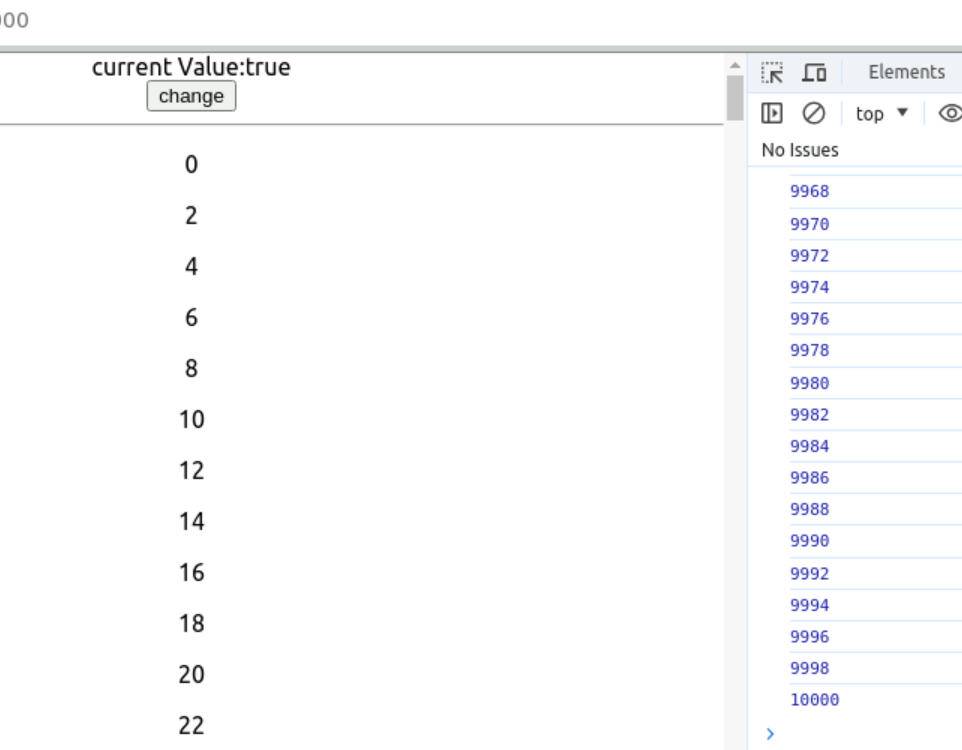
**7.2 npm start**



**7.3 See the output on browser**

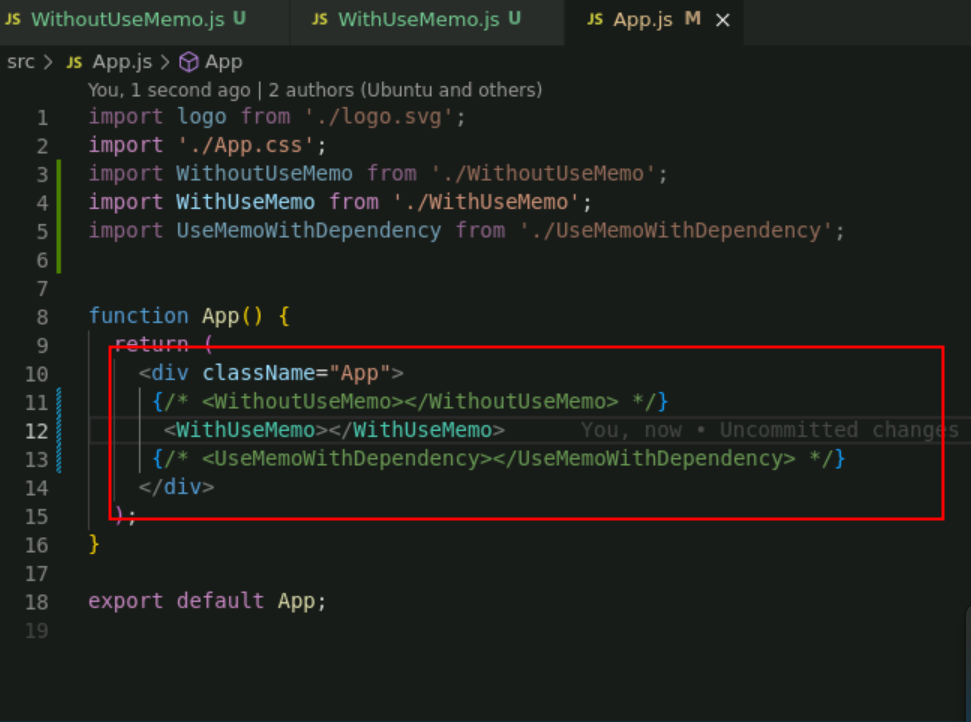


**7.4 when you click button the value change here again the expensive function get called. Which you can check in browser console.**



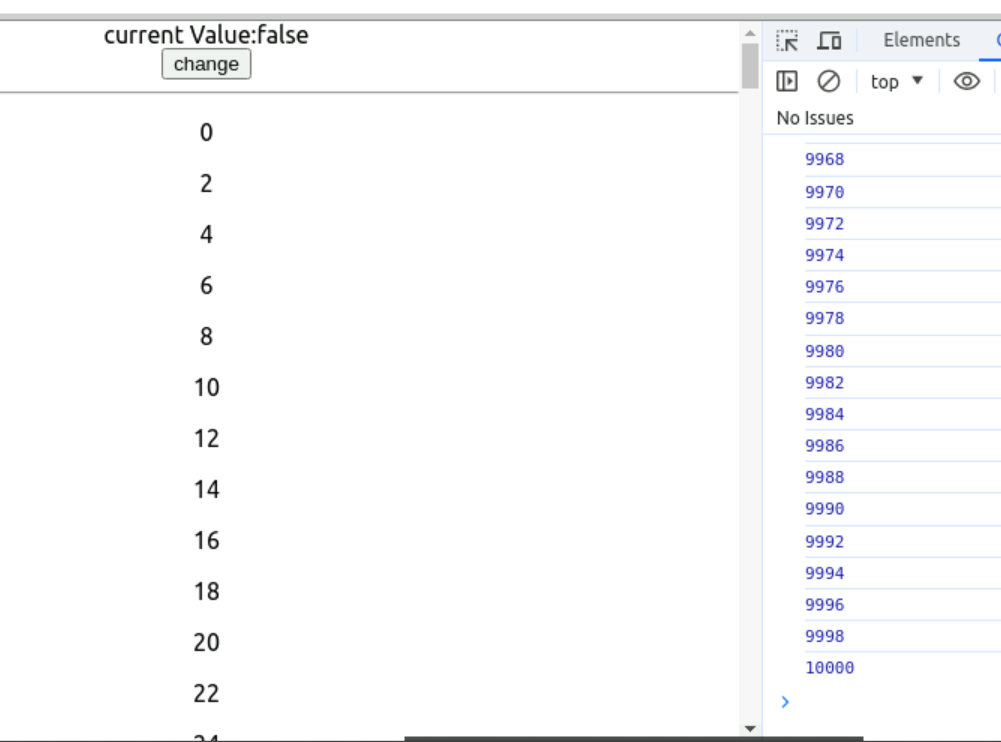
**Step 8 . Testing App.js file with useMemo expensive function call.**

8.1 Do the changes in App.js file now enable use WithUseMemo component

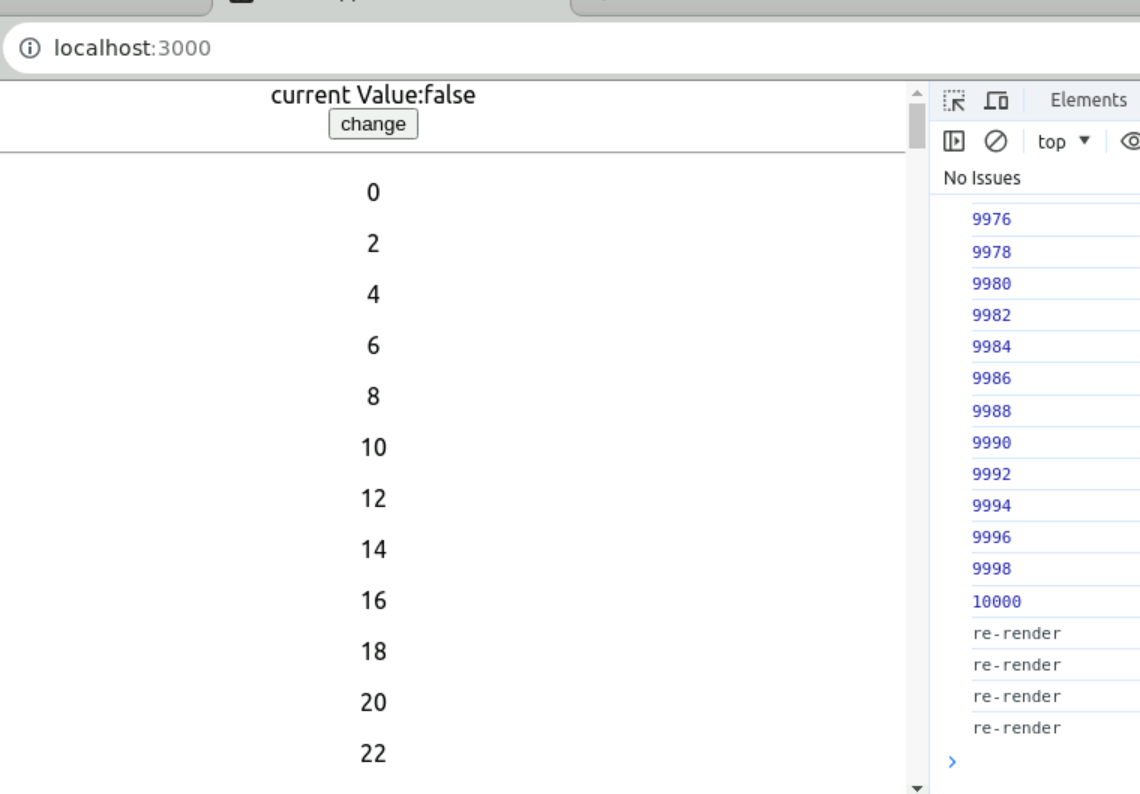


8.2 : Check on browser

First time the expensive function get called.

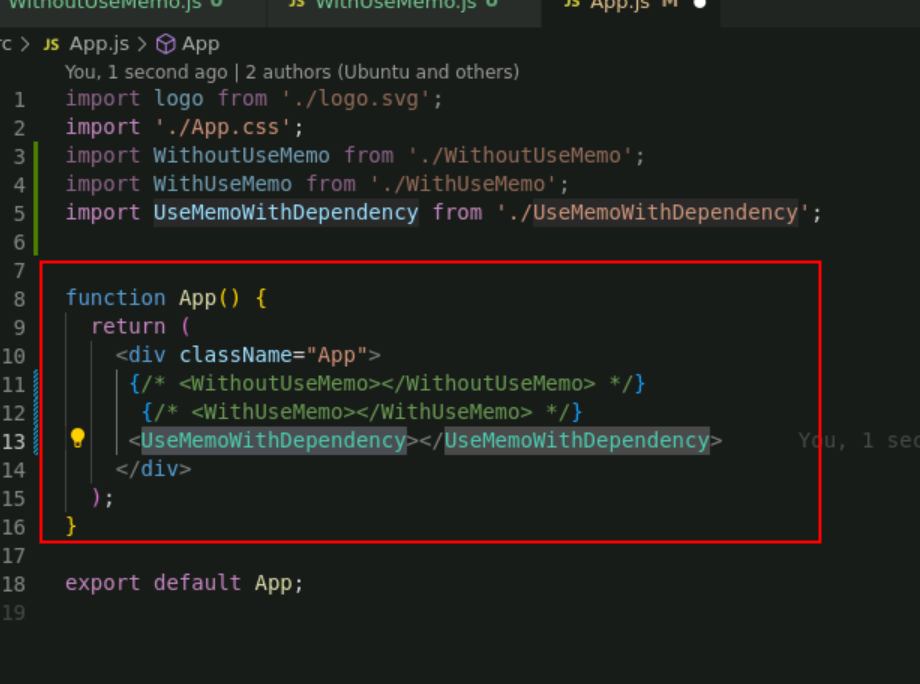


Once you click on button you can check it call only few times and n number of times.

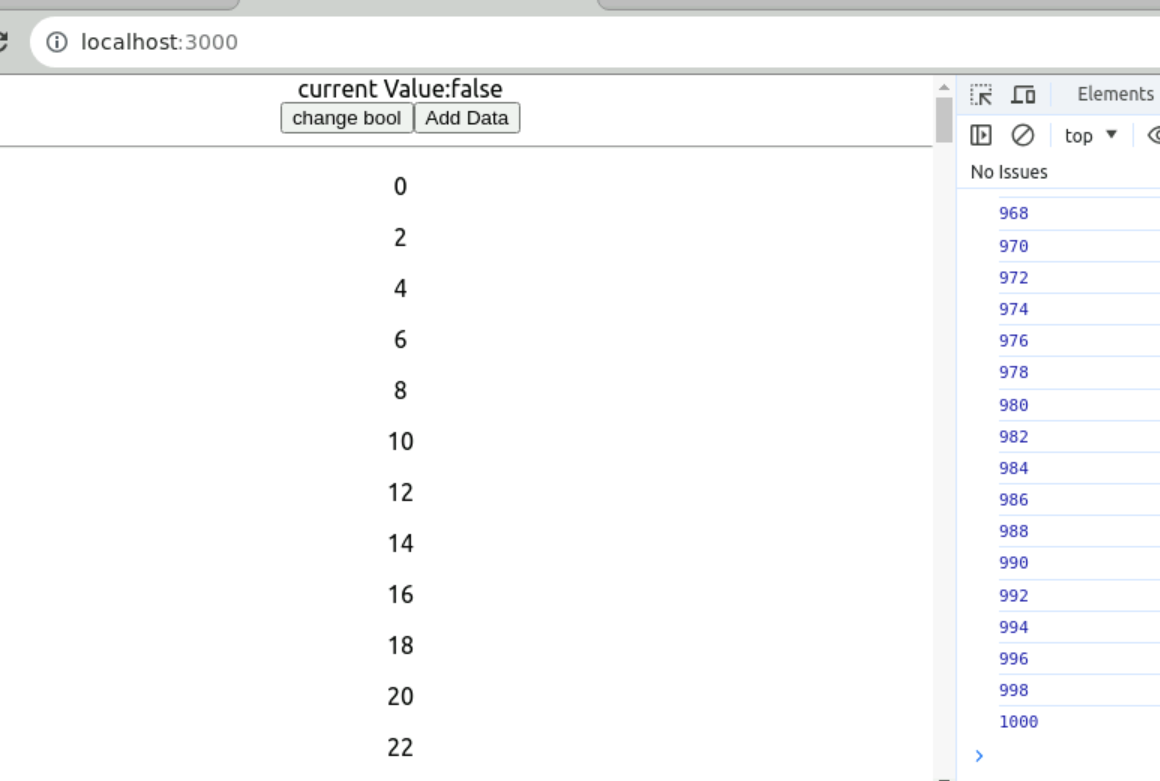


**Step 9. Testing App.js file with useMemo expensive function call with dependencies**

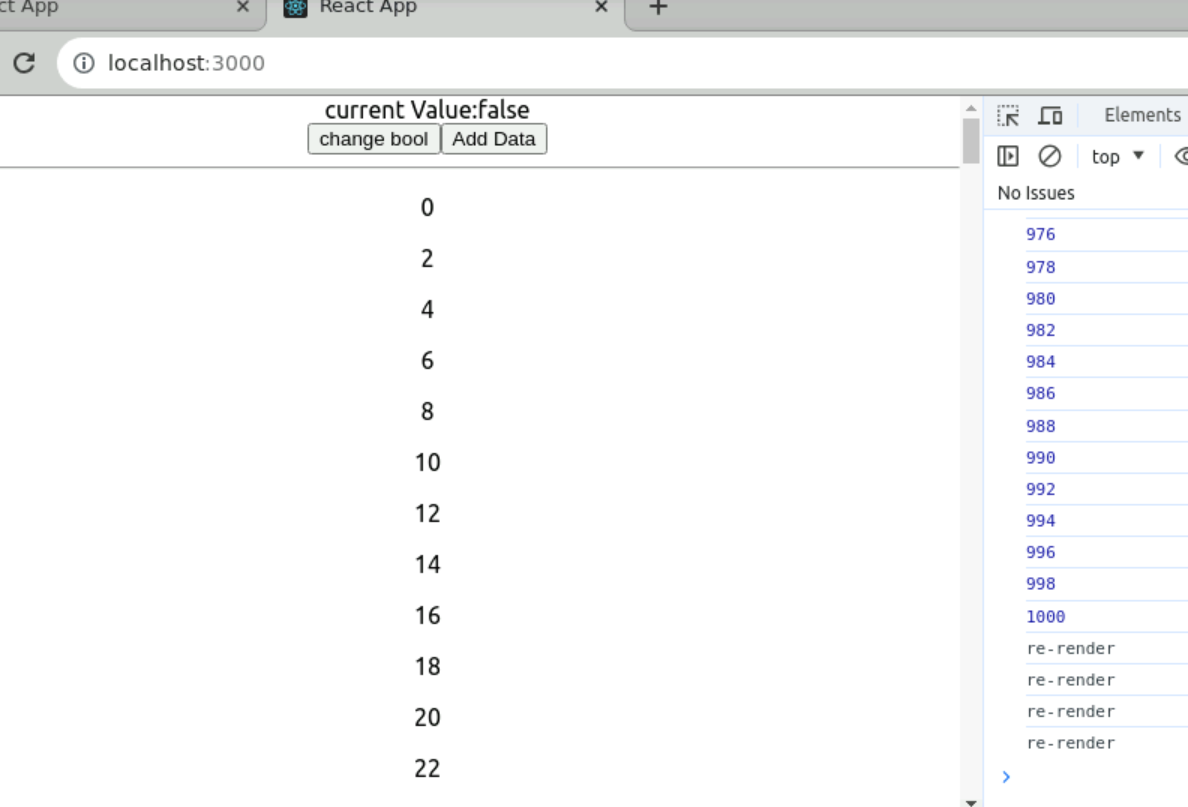
9.1 Do the changes in App.js file now enable use UseMemoWithDependency component



**9.2** Now click on button



If you click on change bool next time it will not call expensive function.



But whenever we call addData function it will call once again function to load all data.

