**Lesson 01 Demo 03**

**Using the useMemo Hook**

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

**Tools required:** Node.js and React.js

**Prerequisites:** None

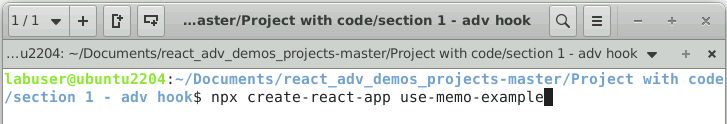
Steps to be followed:

1. Create and set up a React project
2. Perform expensive function calls: without useMemo, with useMemo hook, and with useMemo with dependencies
3. Test the application

**Step 1: Create and set up a React project**

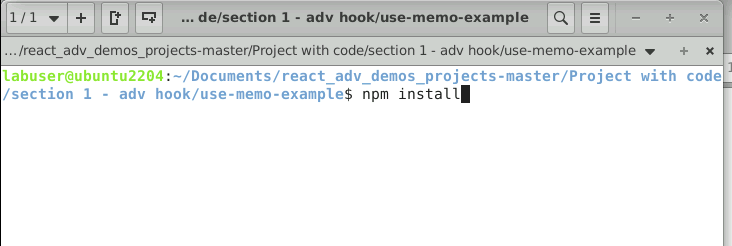
1. Open a terminal window and run the following command to create a React application:

**npx create-react-app use-memo-example**

****

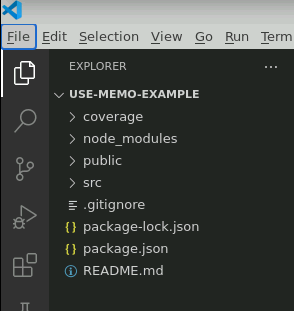
1. Open a terminal window inside a React project, **use-memo-example**, and run the following command to install the required dependencies:

**npm install**

****

**Note:** This command helps install all the required dependencies mentioned in the **package.json** file in the local machine in the form of a **node\_module** folder.

1. Open the **USE-MEMO-EXAMPLE** folder in the VS Code editor



**Step 2: Perform expensive function calls: without useMemo, with useMemo hook, and with useMemo with dependencies**

1. Create the **WithoutUseMemo.js** file with the following code:

**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)=>{**

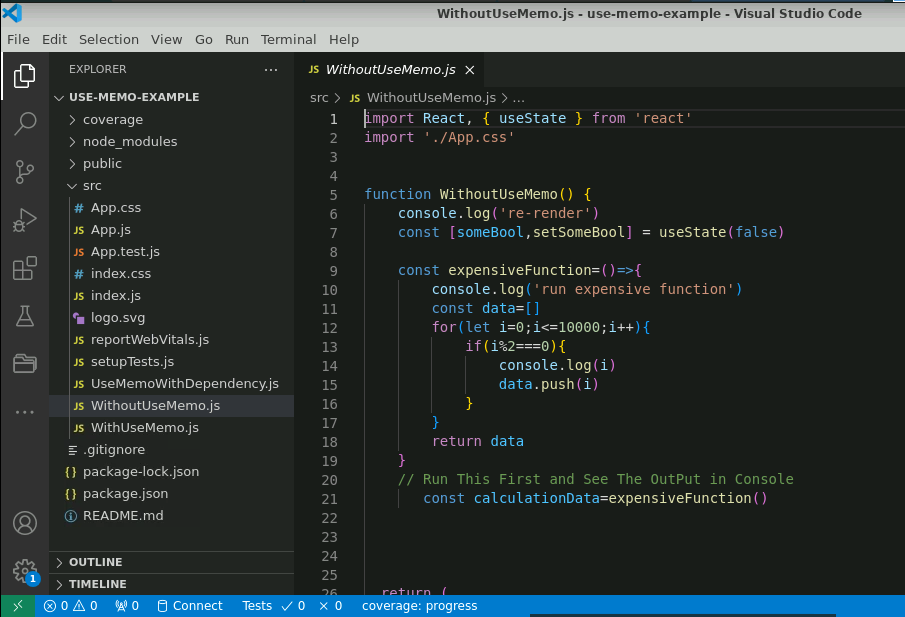
**return <p key={el}>{el}</p>**

**})}**

**</div>**

**}**

**export default WithoutUseMemo;**



**Note:** This component is responsible for executing the loop 10,000 times and adding the value to the data values. This function call is made repeatedly.

1. Create the **WithUseMemo.js** file with the following code:

**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>**

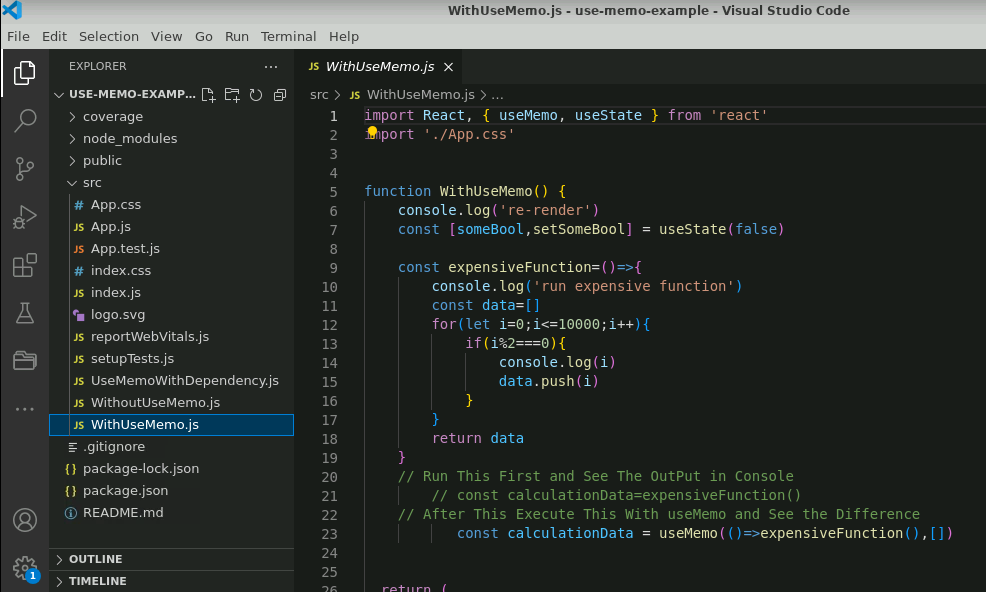
**})}**

**</div>**

**)**

**}**

**export default WithUseMemo;**



**Note:** This component is responsible for executing the loop 10,000 times and adding the value to the data values. This function call is made repeatedly.

1. Create the **UseMemoWithDependency.js** file with the following code:

**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>**

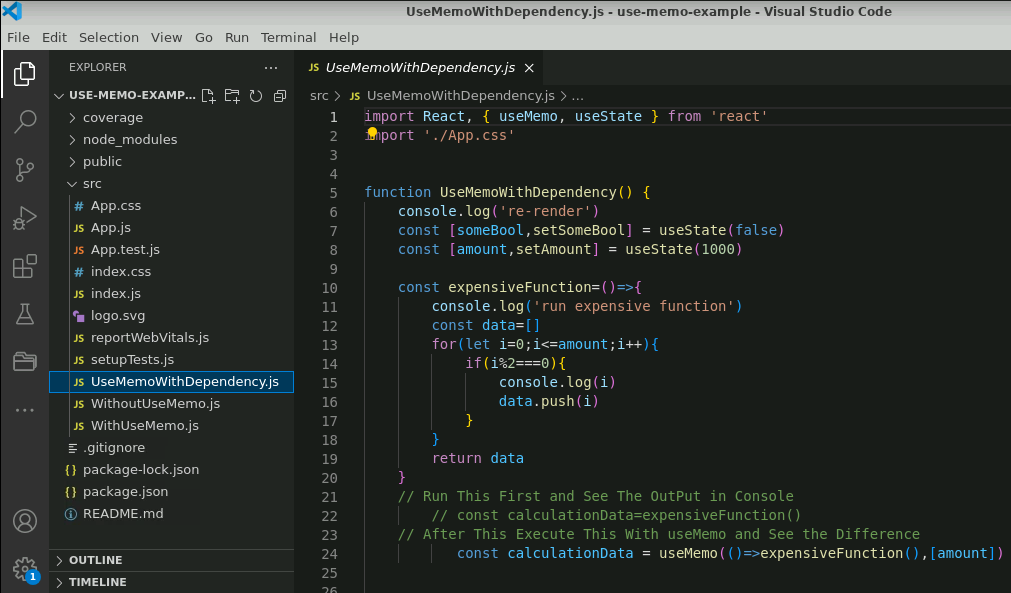
**})}**

**</div>**

**)**

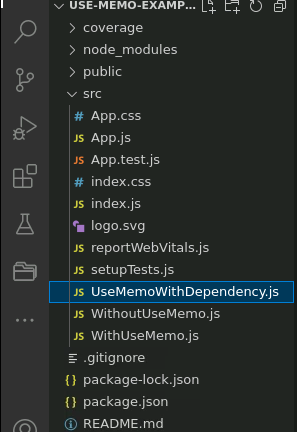
**}**

**export default UseMemoWithDependency;**



**Note:** This component is responsible for executing the loop 10,000 times and adding the value to the data values. This function call is made repeatedly.

1. The following project structure appears as follows:



1. Import all three component files in **App.js**

**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>**

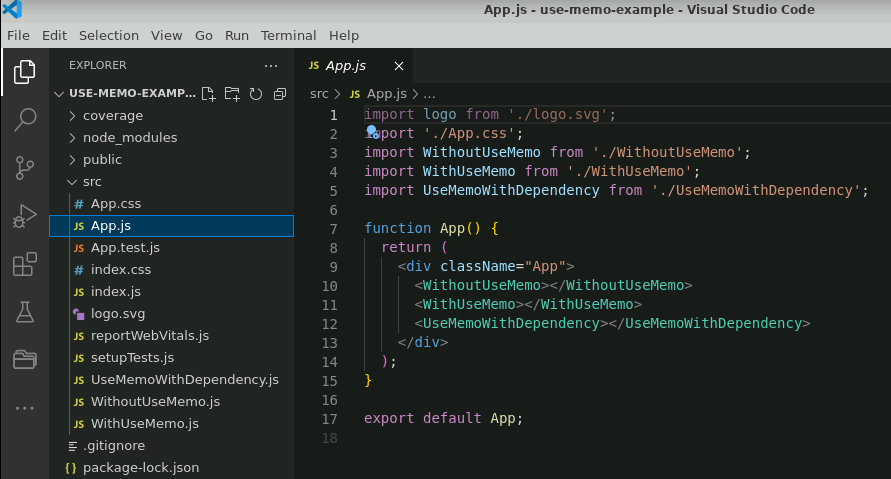
**<WithUseMemo></WithUseMemo>**

**<UseMemoWithDependency></UseMemoWithDependency>**

**</div>**

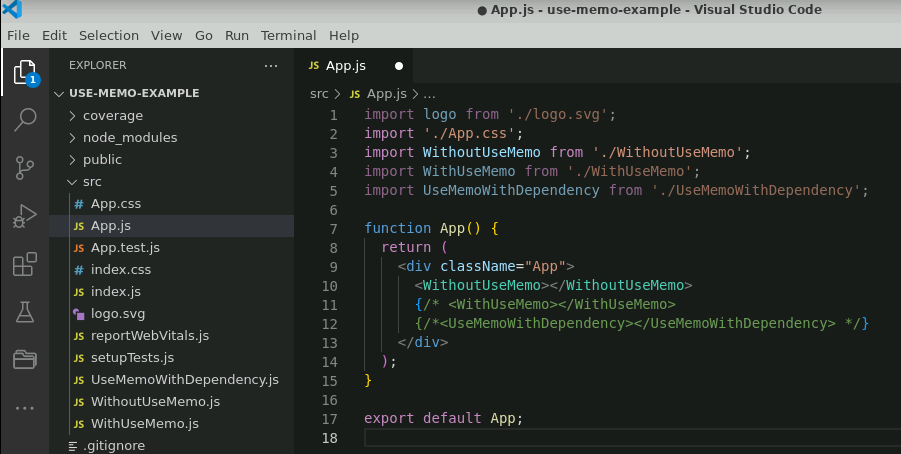
**);}**

**export default App;**



**Step 3: Test the application**

1. Comment the code in the **App.js** file as shown below to test the application without useMemo expensive function call:



1. Run the application using the **npm start** command

A screenshot of a computer program

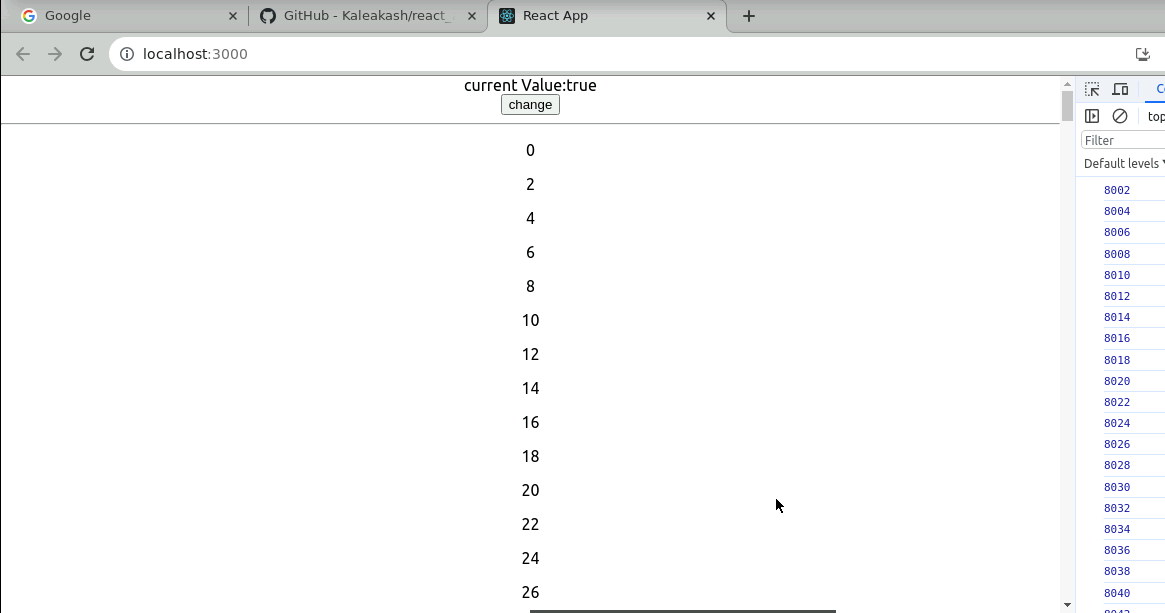
Description automatically generated

After successful execution, the output can be seen as below:

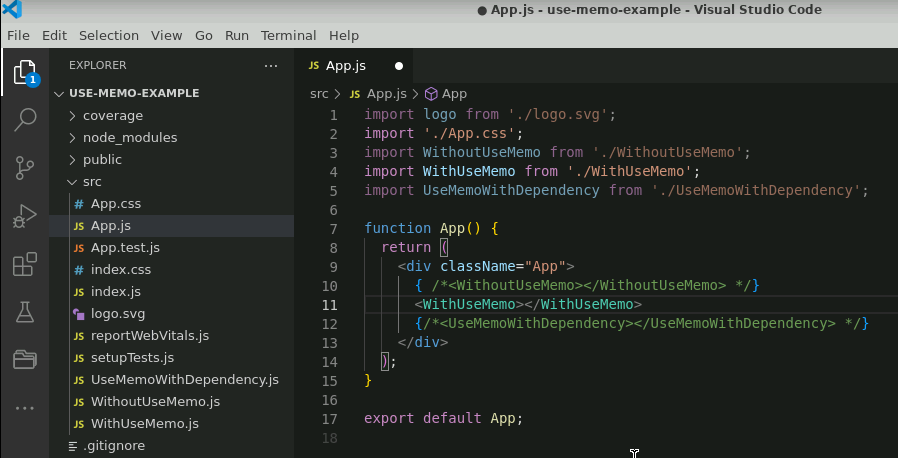
A screenshot of a computer

Description automatically generated

1. Click on the **change** button to update the value and trigger the expensive function



1. Comment the code in the **App.js** file as shown below to test the application with the useMemo expensive function call



1. Run the application using **npm start** command

A screenshot of a computer

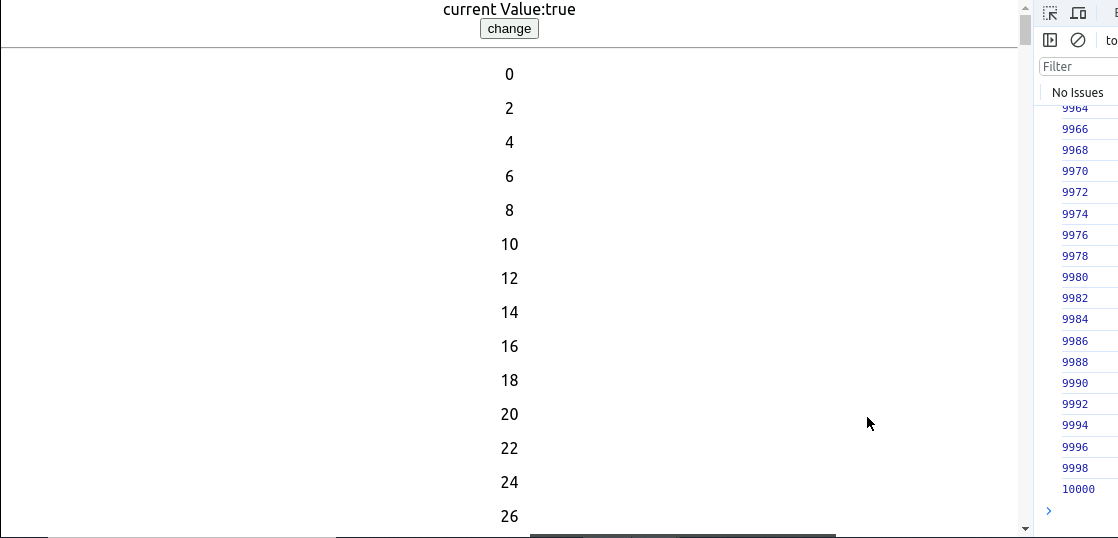
Description automatically generated

After successful execution, the output can be seen as shown below:

A screenshot of a computer

Description automatically generated

1. Click on the change button and see how many times the function is being called



1. Comment the code in the **App.js** file as shown below to test the application with useMemo expensive function call with dependencies

A screen shot of a computer

Description automatically generated

1. Run the application using the **npm start** command

A screenshot of a computer

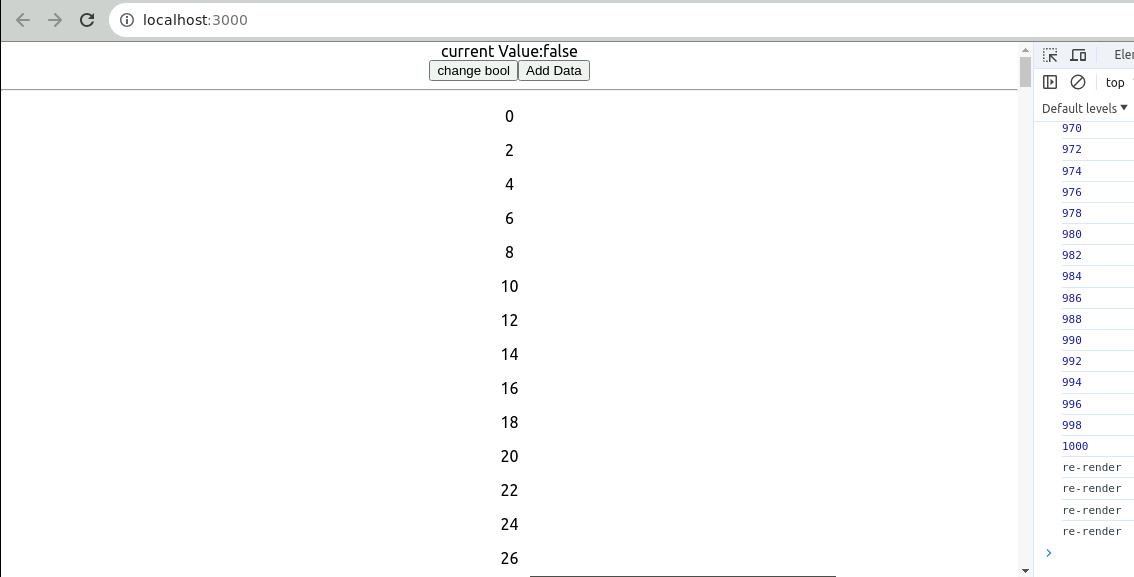
Description automatically generated

After successful execution, the following screen will appear:

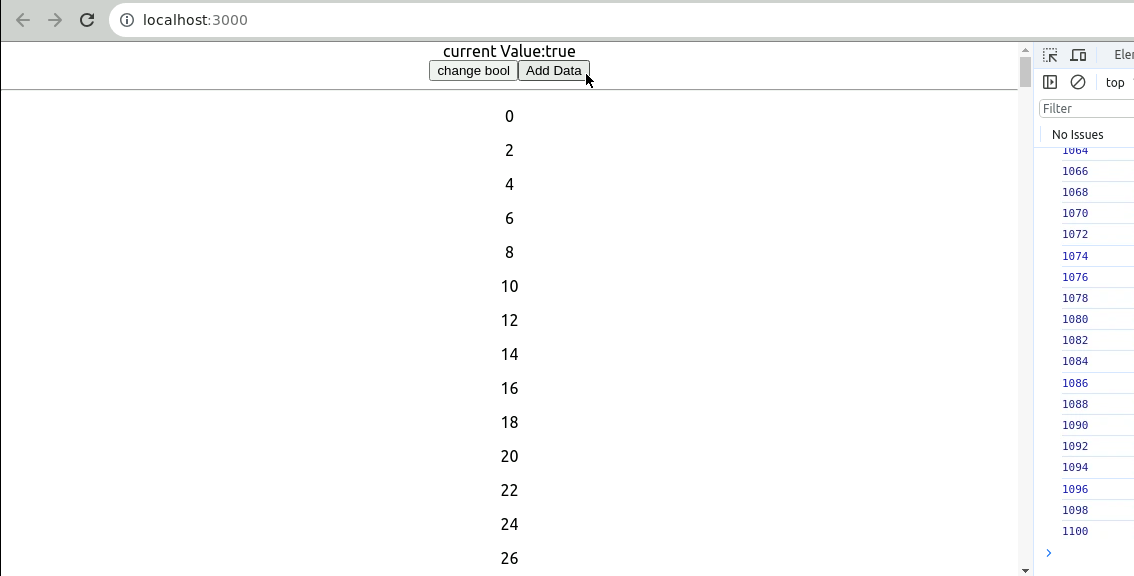
A screenshot of a computer

Description automatically generated

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



1. When the user clicks on the **Add Data** button, it will trigger the function to load all the data once again.



With this, you have successfully showcased the use of the useMemo hook to enhance the performance.