**#. Child Component to Parent Component (using callbacks)**

This one is a bit tricky. We follow the steps below:

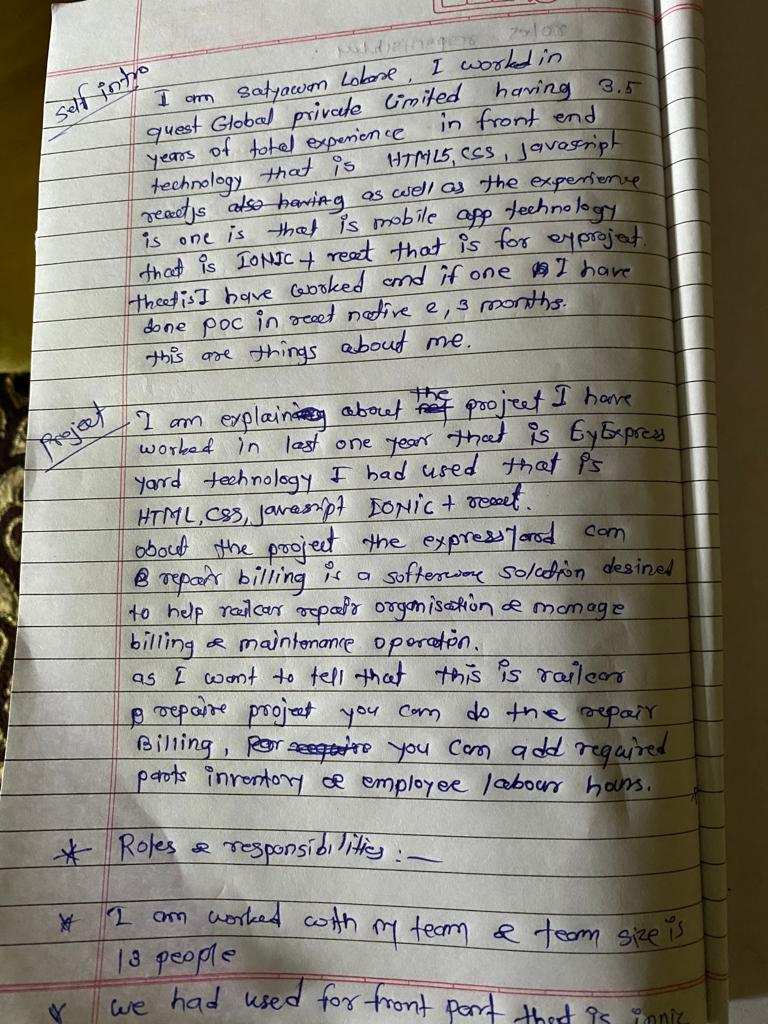
* Create a callback in the parent component which takes in the data needed as a parameter.
* Pass this callback as a prop to the child component.
* Send data from the child component using the callback.

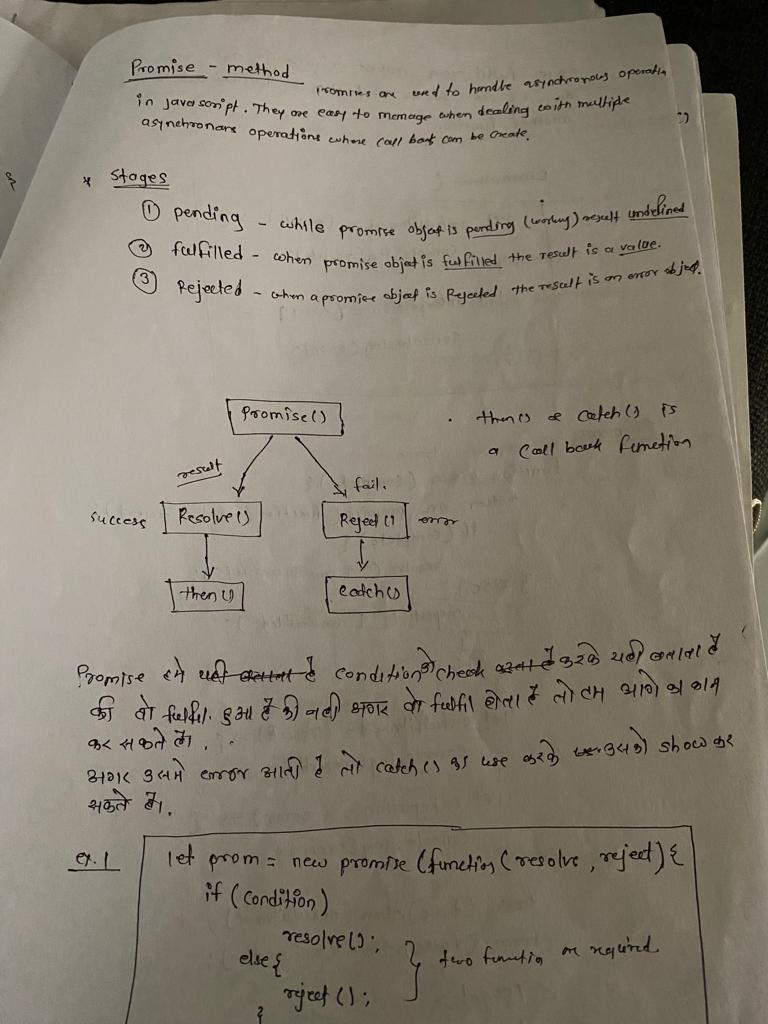
We are considering the same example above but in this case, we are going to pass the updated counterValue from child to parent.

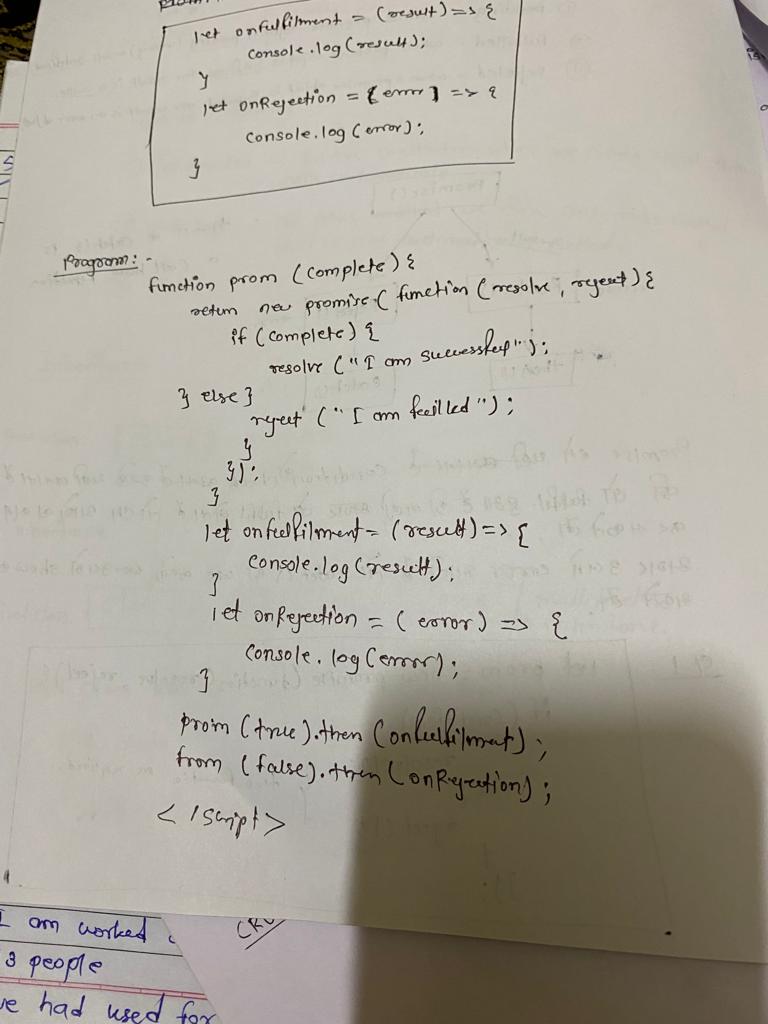
const Parent = () => {  
 const [message, setMessage] = React.useState("Hello World");

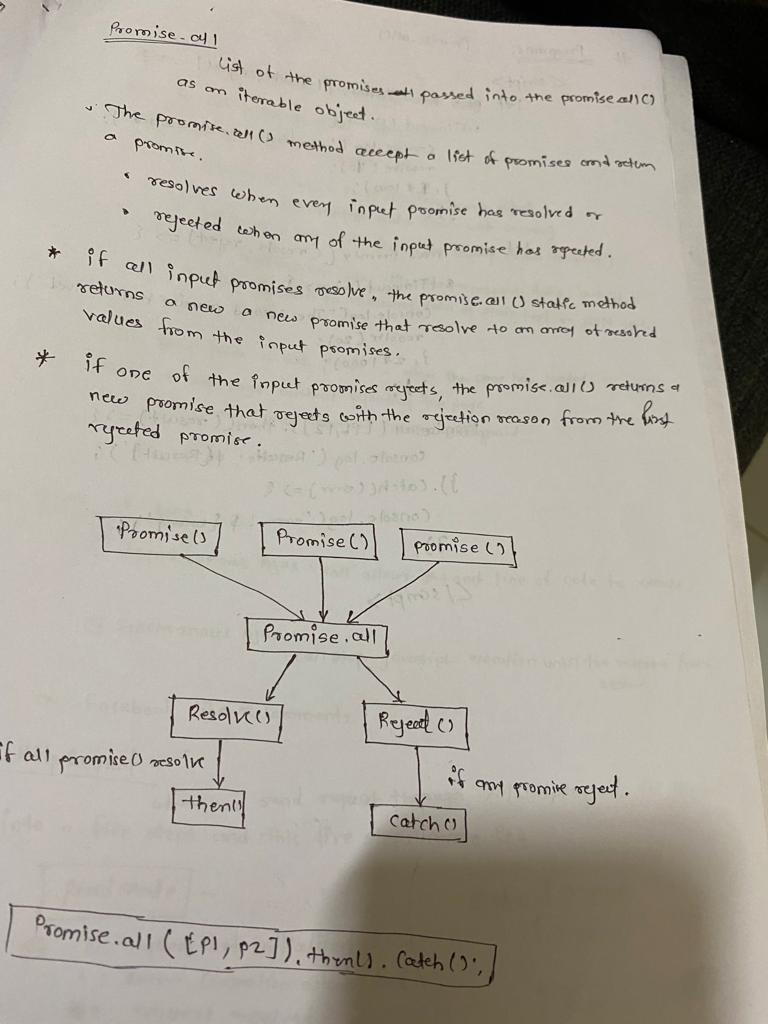
const chooseMessage = (message) => {  
 setMessage(message);  
 }; return (  
 <div>  
 <h1>{message}</h1>  
 <Child chooseMessage={chooseMessage} />  
 </div>  
 );};

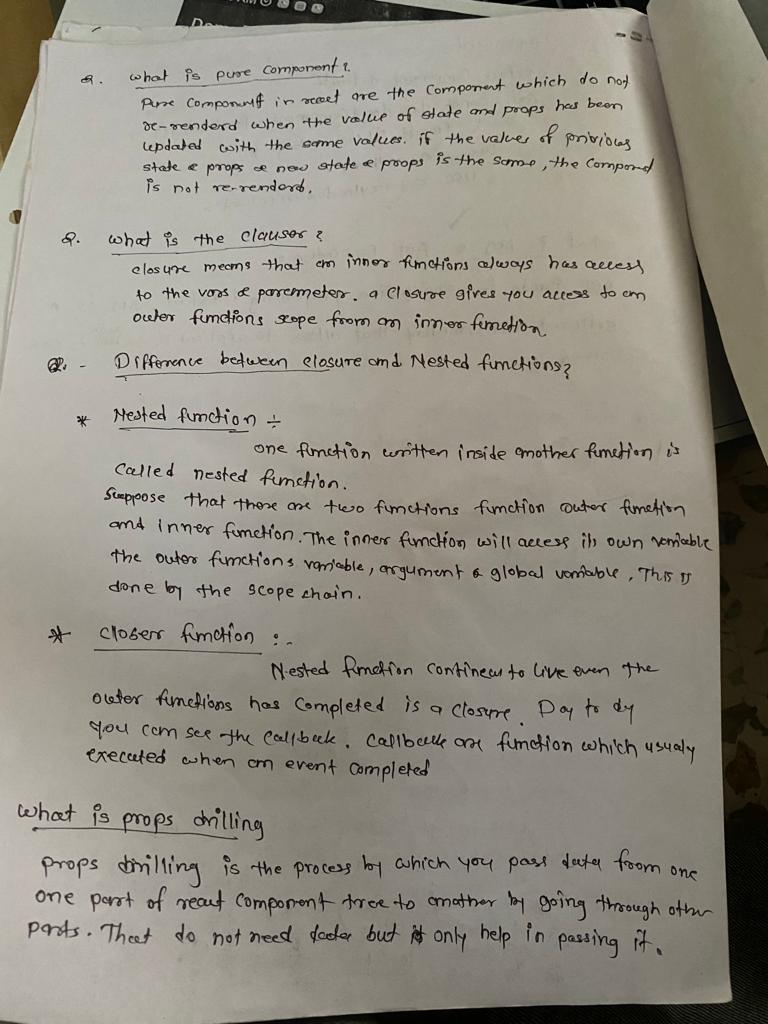
const Child = ({ chooseMessage }) => {  
 let msg = 'Goodbye';  
 return (  
 <div>  
 <button onClick={() => chooseMessage(msg)}>Change Message</button>  
 </div>  
 );};

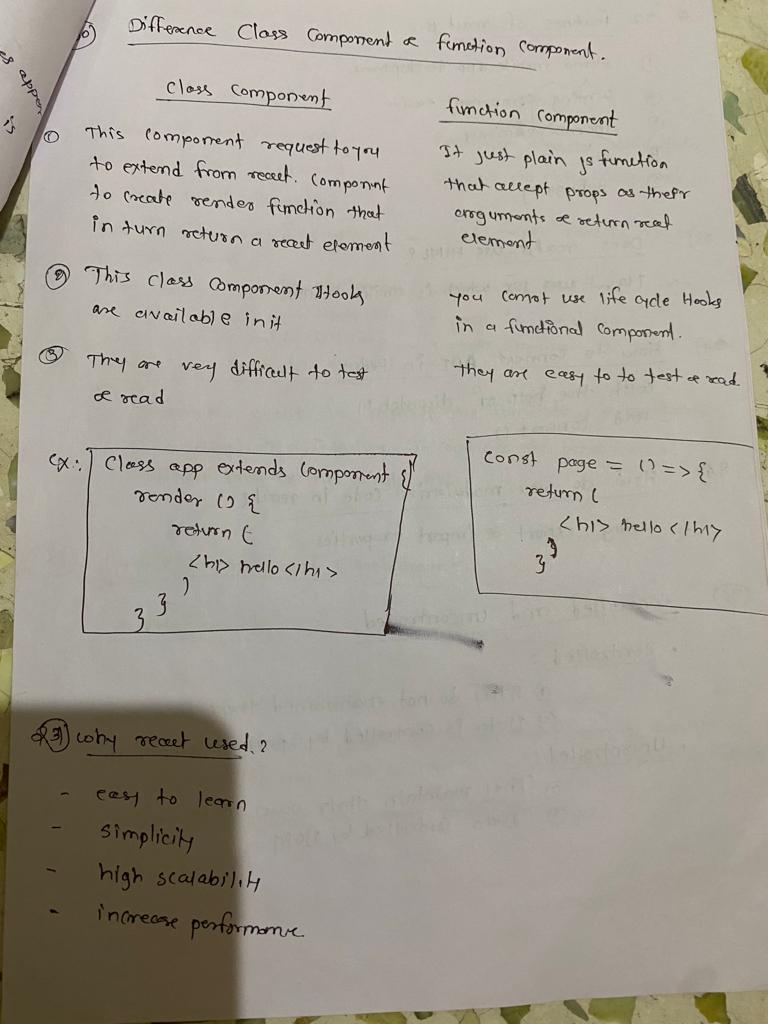












//curring function addition of parenthesis

function global(a){

    return function outer(b){

        return function inner(c){

            console.log(a+b+c);

        }

    }

}

global(1)(2)(3);

output:

6

// reverse string

let string = "Hello how are you?";

let result = string.split("").reverse().join("");

console.log(result);

output:

?uoy era woh olleH

//Reverse string without having same place to words

let string2 = "Hello how are you?";

let result2 = string2.split("").reverse().join("").split(" ").reverse().join(" ");

console.log(result2);

output:

olleH woh era ?uoy

// Multiplication of Nested array using recursive function

const arr = [1, 2,[3,4,[5,6,7]]];

const recursiveMultiplication = arr => {

   let prod = 1;

   for(let i = 0; i < arr.length; i++){

      if(Array.isArray(arr[i])){

         prod \*= recursiveMultiplication(arr[i]);

      }else{

         prod \*= arr[i] || 1;

      };

   };

   return prod;

};

console.log(recursiveMultiplication(arr));

output:

5040

// what is the output

for(let i = 0 ; i < 5; i++){

    setTimeout(()=>{

        console.log(i);

    },1000);

}

Output:

0

1

2

3

4

// what is the out put of the following code

for(var i = 0 ; i < 5; i++){

    setTimeout(()=>{

        console.log(i);

    })

}

Output:

5

5

5

5

5

Q. How we can handle the web api call , while so much data from api coming, like infinite data coming from the backend, how we can handle in front end side?

-> Using Pagination, Infinite scroller, using server side rendering.

Q- what is lexical scoping ?

Q- how we can display numbers after every second?

const counter=(n)=>{

    for(let i=1; i<=n; i++){

        setTimeout(()=>{

            console.log(i)

        },i \* 1000)

    }

}

counter(10);

output:

1 //after one second

2 //after one second

3

4

5

6

7

8

9

10

Cognizant questions below:

Q. To make the deep copy of the nested below object=>

const user = {

        Details : {

            person :{

                "name" :"John"

            }

        }

    }

    const obj = Object.assign({}, user);

   obj.Details.person.name = "lx";

   console.log(obj);

   console.log(user);

   const newObj = JSON.parse(JSON.stringify(user));

   console.log(newObj)

output

{ Details: { person: { name: 'lx' } } }

{ Details: { person: { name: 'lx' } } }

{ Details: { person: { name: 'lx' } } }

Q. remove the duplicate array from given array and sort this array?

**Q.** **get name only pooja using find method(**Find the particular object name from the array of object**)**

 let arr = [{name:"Ketaki"},{name:"Pooja"},{name:"Suhas"},{name:"Pallavi"}]

 let getName = arr.find((item)=>item.name="Pooja")

 console.log(getName);

output:

{ name: 'Pooja' }

**Q. using filter method get conditional data from array**

 const user = [

     {

         name:"Ketaki",

         age:25

     },

     {

         name:"Pooja",

         age:31

     },

     {

         name:"Shweta",

         age:35

     },

     {

         name:"Nikhil",

         age:15

     },

     ]

const result = user.filter((item)=>item.age > 20)

console.log(result)

output:

[

{ name: 'Ketaki', age: 25 },

{ name: 'Pooja', age: 31 },

{ name: 'Shweta', age: 35 }

]

// using reduce function need addition of array

const arr2 = [2,3,4,5,6,7,8,9]

result2= arr2.reduce((x,y)=>x+y);

console.log(result2);

output:44

// using curring function do the addition of parameter

function add(a){

    return function(b){

        return function(c){

            return a+b+c;

        }

    }

}

console.log(add(1)(2)(3));

output: 6

// using map function do perform each array square

let num = [2,3,4,5,6,7,8,9];

let square = num.map((item)=>item\*item);

console.log(square);

output:

[

4, 9, 16, 25,

36, 49, 64, 81

]

Q. Find occurances from given array

const arr = ['a','a','b','c','d','a','b'];

const itemCounter = (value, index) => {

    return value.filter((x) => x == index).length;

};

console.log(itemCounter(arr, 'b')) ;

console.log(itemCounter(arr, 'a'));

output

2

3

Sort method for number and string

const num = [1,3,4,2,88,3,2,4,78,98,54,23];

const sortedArr = num.sort((a,b)=>a-b); //ascending order

console.log("Ascending: ",sortedArr);

const desc2 = num.sort((a,b)=>b-a); //Descending order

console.log("Descending: ",desc2)

//Sorted Ascending order

let tt = [{name:"Ketaki"},{name:"Ashu"},{name:"Chaya"}];

let arrSorted = tt.sort((a,b)=> a.name.localeCompare(b.name));

console.log(arrSorted);

//Sorted Descending order

let tt = [{name:"Ketaki"},{name:"Ashu"},{name:"Chaya"}];

let arrSorted = tt.sort((a,b)=> b.name.localeCompare(a.name));

console.log(arrSorted);

// Example to demonstrate Sorting an array of objects by a string

let students = [

  {

    "name": "Isabella Williams",

    "roll": 38,

  },

  {

    "name": "Sophia Heels",

    "roll": 45,

  },

  {

    "name": "Ava Benjamin",

    "roll": 17,

  },

  {

    "name": "Amelia Brown",

    "roll": 8,

  },

]

// sort by name

students.sort(function(a, b) {

  const nameA = a.name.toUpperCase(); // ignore upper and lowercase

  const nameB = b.name.toUpperCase(); // ignore upper and lowercase

  if (nameA > nameB) {

    return -1;

  }

  if (nameA < nameB) {

    return 1;

  }

  // names must be equal

  return 0;

});

console.log("Students sorted based on the descending order of their names are:")

console.log(students);

//Ascending order:

// sort by name

students.sort(function(a, b) {

  const nameA = a.name.toUpperCase(); // ignore upper and lowercase

  const nameB = b.name.toUpperCase(); // ignore upper and lowercase

  if (nameB > nameA) {

    return -1;

  }

  if (nameB < nameA) {

    return 1;

  }

  // names must be equal

  return 0;

});

console.log("Students sorted based on the Ascending order of their names are:")

Remove duplicate array in array list

let arr2 = ["apple", "mango", "apple",

"orange", "mango", "mango"];

const rev = arr2.filter((item,index) => arr2.indexOf(item) === index);

console.log(rev);

or

second method to remove the the duplicate

let arr = ["apple", "mango", "apple",

          "orange", "mango", "mango"];

function removeDuplicates(arr) {

    return [...new Set(arr)];

}

console.log(removeDuplicates(arr));

Q. without map function need to perform the operations

const arr = [1,2,3];

let result = [];

const muitiply = function(num){

return num\*2; //for multiply 2 for each array element

}

for(let i=0; i<arr.length; i++){

result.push(muitiply(arr[i]));

}

console.log(result);

Q. if having infinite argument passing how we can achive those things

function sum(a) {

return function(b){

if(!b){

return a;

}

return sum(a+b);

}

}

console.log(sum(1)(2)(3)(4)(5)(6)()); //output : 21

Q. const add=(one)=>{

return (...args)=>{

return one+args.reduce((a,b)=>a+b)

}

}

var test = add(1);

console.log(test(2)); //output 3

console.log(test(2,3));//6

console.log(test(4,5,6));// 16

Q. passing data from parent to child and child to parent in react

Name is passed from parent to child. - Ketaki

Surname is passed from child to parent. - Ubale

Parent component- App.

import "./styles.css";

import User from "./User";

export default function App() {

function callback(childData){

return(

<div>{childData}</div>

)

}

return (

<div className="App">

<User name ="Ketaki" handleCallback={callback}/>

</div>

);

}

Child component: User

export default function User(props) {

var name = "Ubale"

return (

<div className="App">

<h1>{props.name}</h1>

<div>

{props.handleCallback(name)}

</div>

</div>

);

}

Debouncing concept using react JS:

import React, { useState } from 'react';

const DebounceExample = () => {

const [searchTerm, setSearchTerm] = useState('');

// State to store the debounced search term

const [debouncedSearchTerm, setDebouncedSearchTerm] = useState('');

// Function to update the search term state

const handleInputChange = (event) => {

setSearchTerm(event.target.value);

};

// Effect to update the debounced search term after a delay

React.useEffect(() => {

// Set a timeout to update debouncedSearchTerm after 500 milliseconds

const debounceTimer = setTimeout(() => {

setDebouncedSearchTerm(searchTerm);

}, 500);

// Cleanup the timer if the component unmounts or searchTerm changes

return () => clearTimeout(debounceTimer);

}, [searchTerm]);

// Perform your search operation using debouncedSearchTerm

return (

<div>

<input

type="text"

placeholder="Search..."

value={searchTerm}

onChange={handleInputChange}

/>

<p>Search term: {searchTerm}</p>

<p>Debounced Search term: {debouncedSearchTerm}</p>

{/\* Render search results or perform other operations \*/}

</div>

);

};

export default DebounceExample;

Q. Using useCallback hook to create button and count the value onclick event. To avoiding the re-rendering.

->

import React, { useCallback } from 'react';

const MyButtonComponent = ({ onClick }) => {

return (

<button onClick={onClick}>

Click me

</button>

);

};

const ParentComponent = () => {

// Define a state variable to track the click count

const [clickCount, setClickCount] = React.useState(0);

// Define a click handler using useCallback

const handleClick = useCallback(() => {

// Increment the click count

setClickCount(clickCount + 1);

// Your other logic here...

}, [clickCount]); // Only recreate the function if clickCount changes

return (

<div>

<p>Click Count: {clickCount}</p>

{/\* Pass the memoized handleClick function to the child component \*/}

<MyButtonComponent onClick={handleClick} />

</div>

);

};

export default ParentComponent;