Day-19

Agenda : Hooks , Form validation , axios (server side)

Tuesday, February 18, 2025 9:05 AM

Functional Components Vs Class Components

Functional Components	Class Components

1.
Function function_name()
{
 	
return(); // html
}
 	
Export default <function_name>
	Import react,{component} from 'react'
	
	Class classname extends Component
	{
	
	Render()
	{
	
	return(); //html
	}
	}Export default <classname>

Simple Functions , apart from this we have some advanced functions as well and we call it as HOOKS	It maintains their own private data - state , here we are using this keyword . It provide lifecycle hooks
stateless	Stateful
Mainly responsible for the UI	Complex to implement UI Logic
Different types of Hooks : 1. useState **Is used to manage the each functional component state** cons[count , setCount] = useState(0); Cons[id , setId] = useState(2); setId +1; 2. useEffect **is used to mount, render or demount the components** 3. useContext **is used for global state management without using redux** Advanced Hooks : 4. useRef 5. useCallback 6. useMemo	setState method to update
Hooks are functions that let you to use State and lifecycle features in functional components without writing a class	

Mount -useEffect(() => {} , [])
Updating / render - useEffect(() => {} ,[dependency])
Unmounting - useEffect(() => { return () => {} ;}, [])

Props & state
Props -- pass an information from one component to another components
State -- It's like a central place for storing the data

So Hook is the new introduction through which we can store the data and it gives state

State is a secret place so if you modify or edit the data in a state we always use setState method for that

useEffect

	CC	FC
Mount	componentDidMount()	useEffect(() => {} , [])
Updating	componentDidUpdate()	useEffect(() => {} ,[dependency])
Unmounting	componentWillUnmount()	useEffect(() => { return () => {} ;}, [])

```
import React ,{useEffect,useState} from "react";
const StockPriceTracker = () => {
  const[stockPrice , setStockPrice] = useState(100); // initial stock Price
```

```
        //first case  -- Mounting when the component is added to the DOM (first renders)
        useEffect(

            () =>    {
                     //mounting
                     console.log("Component Mounted called only once");

                      const interval = setInterval( () => {
                       const newPrice = (Math.random() * (105 - 95) + 95).toFixed(2);
                       console.log(`Stock price Update : ${newPrice}`);
                       setStockPrice(newPrice)  // triggerin a re-ender
                      } , 2000)
                      // unmounting
                      return() => {
                         clearInterval(interval);
                         console.log("Component unmounted")
                      }
                      }, []

                  );

                  return (
                     <div>
                         <h2> Stock Price : {stockPrice} </h2>
                     </div>
                  );
};
export default StockPriceTracker;
```

Optimization

useRef : is use to track the previous search without re-endering. We create a reference to DOM elements or stores values without triggering re-enders . It is also used in accessing DOM elements and storing mutable values

If we want to access the input field without re-endering .

useCallBack() -- Memoizing the functions

Which is used to prevent unnecessary re-creations of the functions in child components

```
import React , {useState ,useCallback} from "react";
const Child  =({handleClick}) => {

    console.log("Button rendered ");
    return <button onClick={handleClick}>Click Here</button>
};
const Parent = () =>{
const [count , setCount] = useState(0);
const memoHandleClick =  useCallback(() => {
  console.log("Button Clicked");
   // setCount((prev) => prev + 1);
}, []);
return (
        <div>
            <h2> Count : {count}</h2>
            <button onClick={() => setCount(count+1)}>Click Here</button>
            <Child handleClick ={memoHandleClick} />
        </div>
)
};
export default Parent;
```