import React, { useEffect } from 'react'

const Cuseeffect = () => {

    useEffect(() => {

        console.log("hello components start");

    }, [])

    return (

        <div>Cuseeffect</div>

    )

}

export default Cuseeffect

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

const Cuseeffect = () => {

    const [count, setCount] = useState(0)

    useEffect(() => {

        console.log("hello components start");

    }, [count])

    return (

        <div>Cuseeffect

            <h4>{count}</h4>

            <button onClick={() => setCount(count + 1)}>Count Button</button>

        </div>

    )

}

export default Cuseeffect

**Why Alert Comes Before UI Updates?**

React asynchronous tarike se state update karta hai. Jab count ki value update hoti hai:

1. State update hone ka signal milta hai.
2. UI re-render hone se pehle useEffect chalta hai (kyunki state ki change track ki ja rahi hai).
3. Pehle alert dikhaya jata hai, uske baad UI update hoti hai.

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

const Cuseeffect = () => {

    const [count, setCount] = useState(0)

    useEffect(() => {

        if (count === 2) {

            alert("hello componets")

        }

    }, [count])

    return (

        <div>Cuseeffect

            <h4>{count}</h4>

            <button onClick={() => setCount(count + 1)}>Count Button</button>

        </div>

    )

}

export default Cuseeffect

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

const Cuseeffect = () => {

    const [count, setCount] = useState(0)

    useEffect(() => {

        if (count === 2) {

            alert("hello componets")

        }

        return () => {

            console.log("end!!!!!!!!!!!!!");

        }

    }, [count])

    return (

        <div>

            Cuseeffect

            <h4>{count}</h4>

            <button onClick={() => setCount(count + 1)}>Count Button</button>

        </div>

    )

}

export default Cuseeffect

import React, { useEffect, useState } from 'react';

export default function ExampleComponent() {

    const [data, setData] = useState(null);

    useEffect(() => {

        fetch('https://jsonplaceholder.typicode.com/todos')

            .then(response => response.json())

            .then(data => setData(data));

    }, []); // Empty dependency array → Runs only once

    return <div>{data ? 'Data Loaded' : 'Loading...'}</div>;

}

import React, { useEffect, useState } from 'react';

export default function ExampleComponent() {

    const [data, setData] = useState(null);

    useEffect(() => {

        fetch('https://jsonplaceholder.typicode.com/todos')

            .then(response => response.json())

            .then(data => setData(data))

            .catch(err => console.log(err.message));

    }, []);

    return (

        <div>

            {data ? <>

                {

                    data.map((item, index) => (

                        <div key={index}>

                            <div>

                                <h3>User Id: <span style={{ fontWeight: "lighter" }}>{item.id}</span> </h3>

                                <h3>Title: <span style={{ fontWeight: "lighter" }}> {item.title}</span></h3>

                            </div>

                        </div>

                    ))

                }

            </> : 'Loading...'}

        </div>

    );

}

React ka **Strict Mode** ek tool hai jo development ke doran aapki app ki errors aur potential issues ko identify karne mein madad karta hai. Yeh sirf development environment mein kaam karta hai aur production build mein iska koi asar nahi hota.

Yeh kuch kaam karta hai:

1. **Warnings Show Karna**: Strict Mode un practices ya codes par warning deta hai jo React ke future updates ke sath compatible nahi ho sakte, jaise deprecated lifecycle methods.
2. **Unintentional Side Effects Pakarna**: Yeh lifecycle methods ke ander unintended side effects ka pata lagane ke liye extra renders perform karta hai, jo aapke code ko zyada robust banata hai.
3. **Legacy Features Disable Karna**: Purani aur unsafe features ko discourage karta hai, taake code modern aur secure rahe.

Example:

import React from 'react';

import ReactDOM from 'react-dom';

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>

);

Iska faida yeh hai ke yeh aapke code ko improve karne aur future-proof banane mein help karta hai. Lekin yeh sirf developer experience ke liye hai aur user experience par koi direct asar nahi dalta.