

ClassCounter.js

import React, { Component } from 'react';

class ClassCounter extends Component {

    constructor(props) {

        super(props);

        this.state = {

            count:0

        }

    }

    increamentCounter = () =>{

        this.setState(prevState => {

            return {count:prevState.count + 1}

        })

    }

    render() {

        const {count} = this.state

        return (

            <div>

                <button onClick = {this.increamentCounter}>Click {count}</button>

            </div>

        );

    }

}

export default ClassCounter;

HookCounter.js

import React, {useState} from 'react';

function HookCounter(props) {

    const [count,setState] = useState(0) //destructuring array syntax

    return (

        <div>

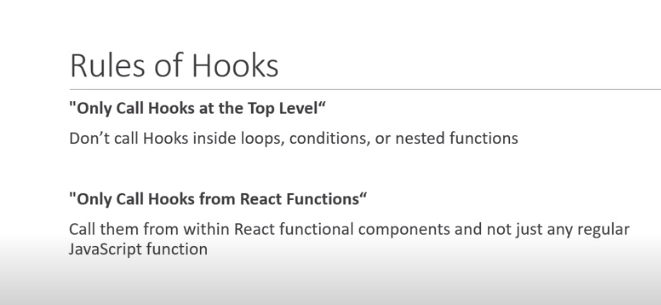
            <button onClick = {() => setState(count + 1)}>Click from hook - {count}</button>

        </div>

    );

}

export default HookCounter;



HookCounterTwo.js : with using prevState or prevCount

import React,{useState} from 'react';

function HookCounterTwo(props) {

    const initialCount = 0;

    const [count,setCount] = useState(0)

    const increamentFive = () => {

        for(let i=0;i<5;i++){

           // setCount(count + 1)//while using this to increament count by 5 so it will increame wid one only so we need to use prevCOunt here this is correct approach to use

            setCount(prevCount => prevCount + 1)

        }

    }

    return (

        <div>

           <h1> Count Value - {count}</h1><hr/>

            <button className = "btn" onClick = {() => setCount(prevCount => prevCount + 1)}>Increament Count</button>&nbsp;

            <button onClick = {() => setCount(prevCount => prevCount - 1)}>Decreament Count</button>&nbsp;

            <button onClick = {() => setCount(initialCount)}>Reset Count</button>&nbsp;

            <button onClick = {increamentFive}>Increament by Five</button>

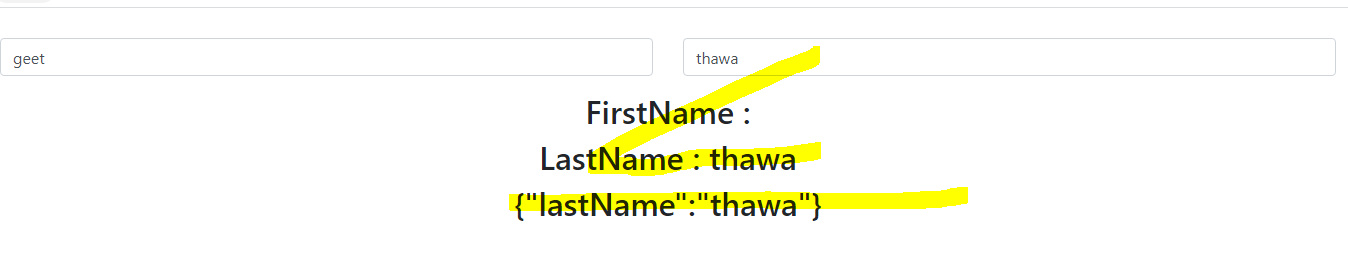
        </div>

    );

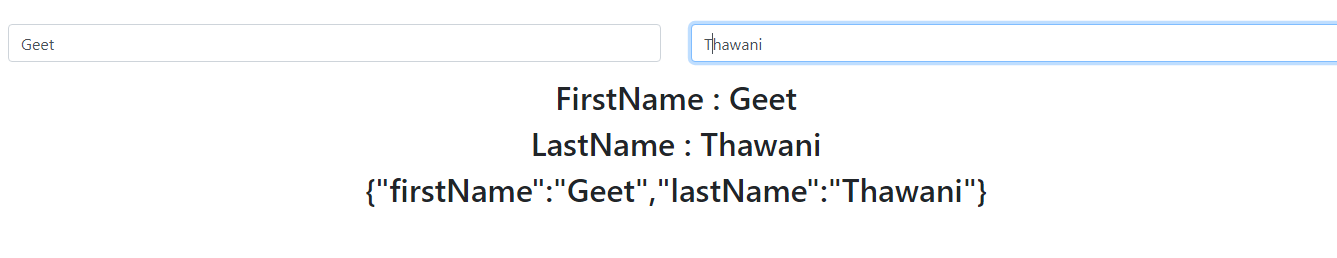
}

export default HookCounterTwo;

useState : It does merege automatically or update any object or array, we have to merge by itself then pass the value we can merge value using spread operator.(…name)



After using spread operator : …name



HookCounterThree.js

import React, {useState} from 'react';

function HookCounterThree(props) {

    const [name,setName] = useState({firstName:'',lastName:''})

    return (

        <div className = 'container-fluid' style={{marginTop:'30px'}}>

            <form className = "form-group row">

                <div className = 'col'>

                    <input

                    type="text"

                    value = {name.firstName}

                    onChange = {(e) => setName({...name, firstName:e.target.value})} className = "form-control"/>

                </div>

                <div className = 'col'>

                    <input

                    type="text"

                    value = {name.lastName}

                    onChange = {(e) => setName({...name, lastName:e.target.value})} className = "form-control"/>

                </div>

            </form>

            <h2>FirstName : {name.firstName}</h2>

            <h2>LastName : {name.lastName}</h2>

            <h2>{JSON.stringify(name)}</h2>

        </div>

    );

}

export default HookCounterThree;

HookCounterArr.js

import React,{useState} from 'react';

function HookCounterArr(props) {

    const[items,setItems] = useState([])

    const addItems = () =>{

        setItems([...items,{

            id:items.length,

            value:Math.floor(Math.random() \* 10 + 1)

        }])

    }

    return (

        <div>

            <button onClick = {addItems}>Add Random Num</button>

            <ul>

                {

                    items.map(item => <li key = {item.id}>{item.value}</li>)

                }

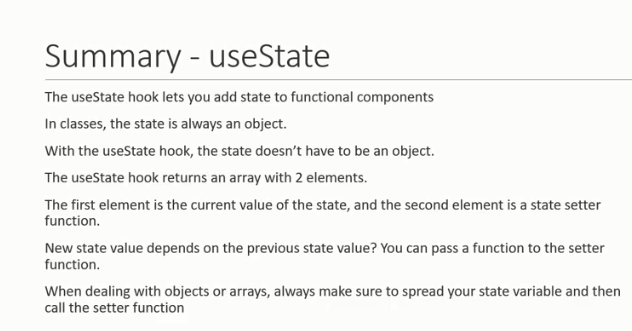
            </ul>

        </div>

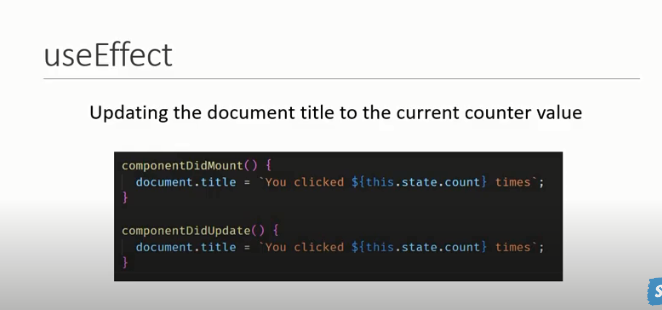
    );

}

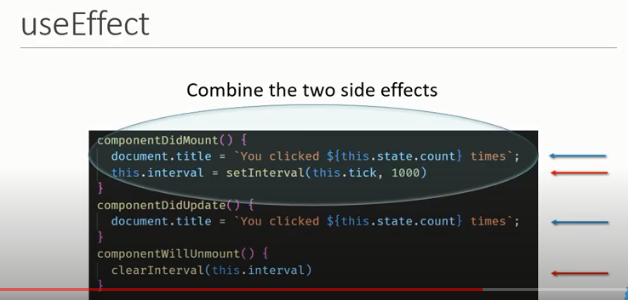
export default HookCounterArr;

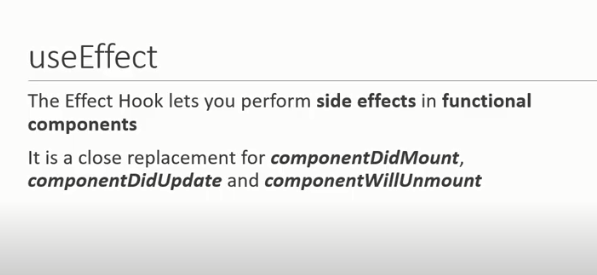


useEffect:









ClickCounterOne.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class ClickCounterOne extends Component {

    constructor(props) {

        super(props);

        this.state = {

            count:0

        }

    }

    componentDidMount() {

        document.title = `Clicked count ${this.state.count}`;

    }

    componentDidUpdate() {

        document.title = `Clicked count ${this.state.count}`;

    }

    render() {

        const {count} = this.state

        return (

            <div>

                <button onClick = {() => this.setState({count:this.state.count + 1})}>Click Count {count}</button>

            </div>

        );

    }

}

export default ClickCounterOne;

HookCOunterEffect.js

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

function HookCounterEffect(props) {

    const [count,setCount] = useState(0)

    useEffect(() => {

        document.title = `You clicked ${count} times`

    })

    return (

        <div>

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

        </div>

    );

}

export default HookCounterEffect;

ClickCounterOne.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class ClickCounterOne extends Component {

    constructor(props) {

        super(props);

        this.state = {

            count:0,

            name:''

        }

    }

    componentDidMount() {

        document.title = `Clicked count ${this.state.count}`;

    }

    componentDidUpdate(prevProps,prevState) {

        //alert(prevState.count +'\_\_\_'+this.state.count);

        if(prevState.count !== this.state.count)

            console.log('update count numer...');

        document.title = `Clicked count ${this.state.count}`;

    }

    render() {

        const {count,name} = this.state

        return (

            <div>

                <input type = 'text' value = {name} onChange = {(e) => this.setState({name:e.target.value})}/>

                <button onClick = {() => this.setState({count:this.state.count + 1})}>Click Count {count}</button>

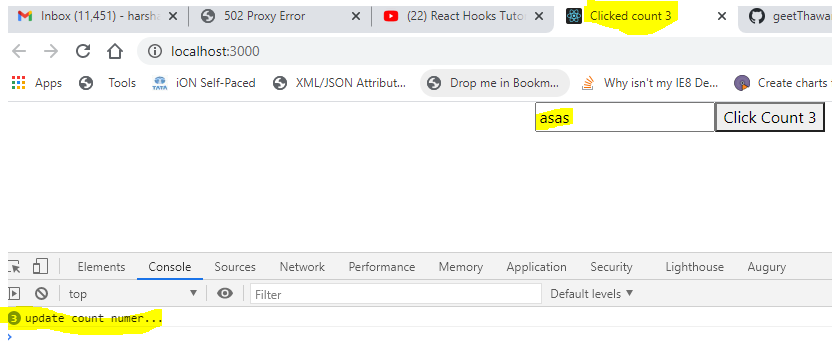
            </div>

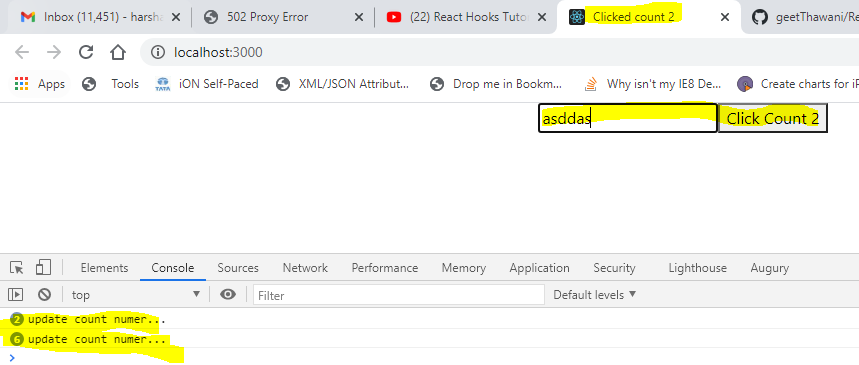
        );

    }

}

export default ClickCounterOne;





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

function HookCounterEffect(props) {

    const [count,setCount] = useState(0)

    const[name,setName] =  useState('')

    useEffect(() => {

        console.log('calling effect fn...');

        document.title = `You clicked ${count} times`

    },[count])//while passing second parameter as an array usinf useEffect to focus only count value while it update then only call useeffect fn.

    return (

        <div>

            <input type = "text" value = {name} onChange = {e => setName(e.target.value)}/>

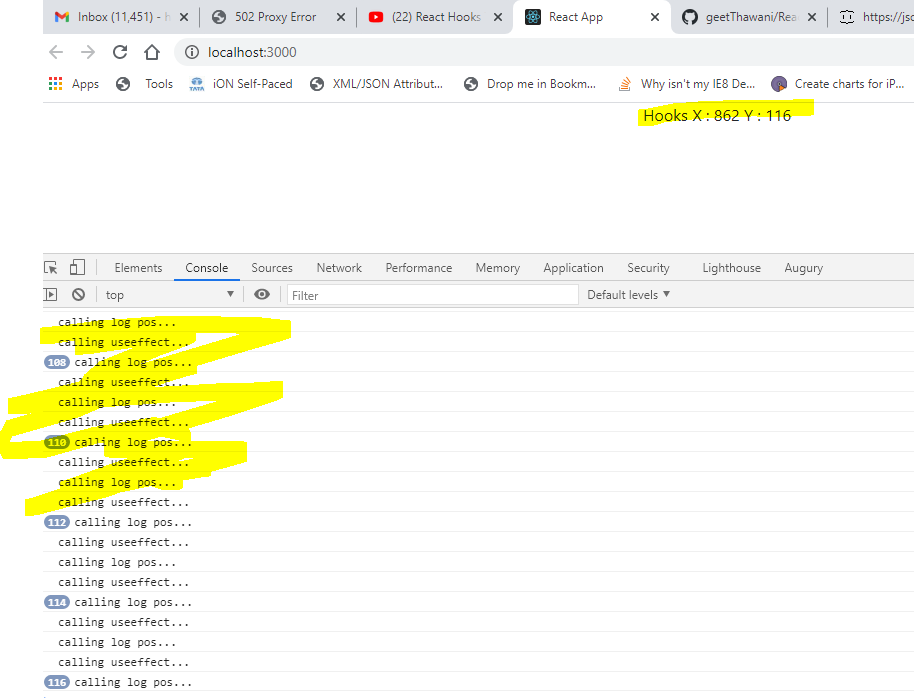
            <button onClick = {() => setCount(count + 1)}>Click Count {count} times</button>

        </div>

    );

}

export default HookCounterEffect;



HookMouse.js

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

function HookMouse(props) {

    const [x,setX] = useState(0)

    const [y,setY] = useState(0)

    const logMousePos = (e) =>{

        console.log('calling log pos...');

        setX(e.clientX)

        setY(e.clientY)

    }

    useEffect(() => {

        console.log('calling useeffect...');

        window.addEventListener('mousemove',logMousePos)

    },[])

    return (

        <div>

            Hooks X : {x} Y : {y}

        </div>

    );

}

export default HookMouse;

ClassMouse.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class ClassMouse extends Component {

    constructor(props) {

        super(props);

        this.state = {

            x:0,

            y:0

        }

    }

    logMousePos = (e) => {

        this.setState({

            x:e.clientX,

            y:e.clientY

        })

    }

    componentDidMount() {

        console.log('log position...');

        window.addEventListener('mousemove',this.logMousePos)

    }

    render() {

        const{x,y} = this.state

        return (

            <div>

                Class x : {x} Y : {y}

            </div>

        );

    }

}

ClassMouse.propTypes = {

};

export default ClassMouse;

UseEffect with cleanupcode.

import React,{useState} from 'react';

import HookMouse from './HookMouse';

function MouseContainer(props) {

    const[display,setDisplay] = useState(true)

    return (

        <div>

            <button onClick = {() => setDisplay(!display)}>Toggle Display </button>

            {display && <HookMouse />}

        </div>

    );

}

export default MouseContainer;

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

function HookMouse(props) {

    const [x,setX] = useState(0)

    const [y,setY] = useState(0)

    const logMousePos = (e) =>{

        console.log('calling log pos...');

        setX(e.clientX)

        setY(e.clientY)

    }

    useEffect(() => {

        console.log('calling useeffect...');

        window.addEventListener('mousemove',logMousePos)

        return ()=>{ //while using this for unmount our event

            console.log('hook component unmounting');

            window.removeEventListener('mousemove',logMousePos)

        }

    },[])

    return (

        <div>

            Hooks X : {x} Y : {y}

        </div>

    );

}

export default HookMouse;

UseEffect with incorrect dependency:

IntervalClassCounter.js

import React, { Component } from 'react';

class IntervalClassCounter extends Component {

    constructor(props) {

        super(props);

        this.state = {

            count:0

        }

    }

    componentDidMount() {

        this.interval = setInterval(this.tick,1000)

    }

    componentWillUnmount() {

        clearInterval(this.interval)

    }

    tick = () => {

        this.setState({

            count:this.state.count + 1

        })

    }

    render() {

        const {count} = this.state

        return (

            <div>

               <h2> Display interval from class component - {count} </h2>

            </div>

        );

    }

}

export default IntervalClassCounter;

HookIntervalCounter.js

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

function HookIntervalCounter(props) {

    const [count,setCount] = useState(0)

    useEffect(() => {

        const interval =  setInterval(tick,1000);

    },[])//if we are not using prevState so we should pass Count but always use prevstate while doing increament

    const tick = () =>{

        // const interval = setCount(count + 1)//pls dont use this appraoch

        const interval =  setCount(prevCount => prevCount + 1)//like this approach is always correct

        return () => {

            clearInterval(interval)

        }

    }

    return (

        <div>

           <h2> Count from Hook {count}</h2>

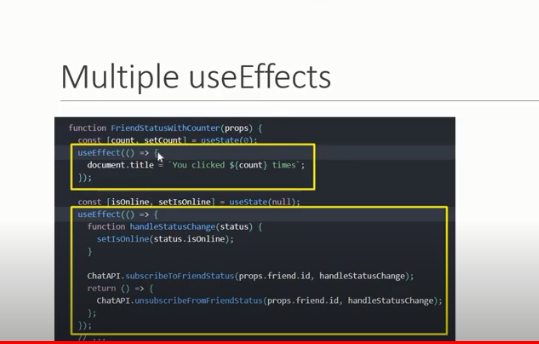
        </div>

    );

}

export default HookIntervalCounter;

We can use multiple useEffect in singlecomponent.



DataFetching.js

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

import axios from 'axios'

function DataFetching(props) {

    const [posts,setPosts] = useState([])

    useEffect(() => {

        axios.get('https://jsonplaceholder.typicode.com/posts')

        .then(res => {

            console.log(res);

            setPosts(res.data)

        })

        .catch(err => {

            console.log(err);

        })

    },[])//pass blank paramaeter so it will call only once a time

    return (

        <div>

            <ul>

                {

                    posts.map(post => <li key = {post.id}>{post.title}</li>)

                }

            </ul>

        </div>

    );

}

export default DataFetching;

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

import axios from 'axios'

function DataFetching(props) {

    // const [posts,setPosts] = useState([])

    const [post,setPost] = useState({})

    const [id,setId] = useState(1)

    useEffect(() => {

        axios.get(`https://jsonplaceholder.typicode.com/posts/${id}`)

        .then(res => {

            console.log(res);

            setPost(res.data)

        })

        .catch(err => {

            console.log(err);

        })

    },[id])//pass blank paramaeter so it will call only once a time or else pass dependency parameter like id

    return (

        <div>

            {/\* <ul>

                {

                    posts.map(post => <li key = {post.id}>{post.title}</li>)

                }

            </ul> \*/}

            <input type = 'text' value = {id} onChange = {(e) => setId(e.target.value)} />

            <h3>{post.title}</h3>

        </div>

    );

}

export default DataFetching;

Fetching data with button click :

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

import axios from 'axios'

function DataFetching(props) {

    // const [posts,setPosts] = useState([])

    const [post,setPost] = useState({})

    const [id,setId] = useState(1)

    const [idFromBtnClk,setIdFromBtnClk] = useState(1)

    useEffect(() => {

        axios.get(`https://jsonplaceholder.typicode.com/posts/${idFromBtnClk}`)

        .then(res => {

            console.log(res);

            // setPosts(res.data)

            setPost(res.data)

        })

        .catch(err => {

            console.log(err);

        })

    },[idFromBtnClk])//pass blank paramaeter so it will call only once a time or else pass dependency parameter like id

    return (

        <div>

            {/\* <ul>

                {

                    posts.map(post => <li key = {post.id}>{post.title}</li>)

                }

            </ul> \*/}

            <input type = 'text' value = {id} onChange = {(e) => setId(e.target.value)} />

            <button onClick  = {() => setIdFromBtnClk(id)}>Fetch post</button>

            <h3>{post.title}</h3>

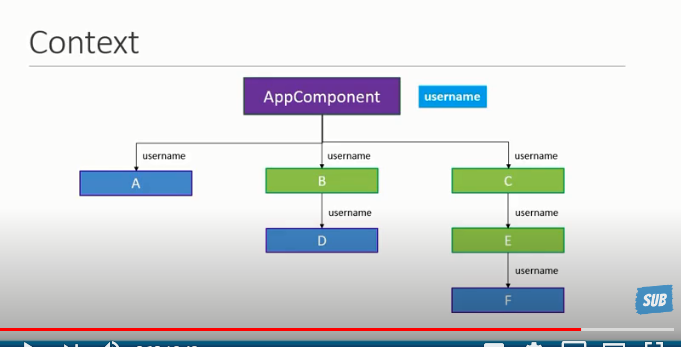
        </div>

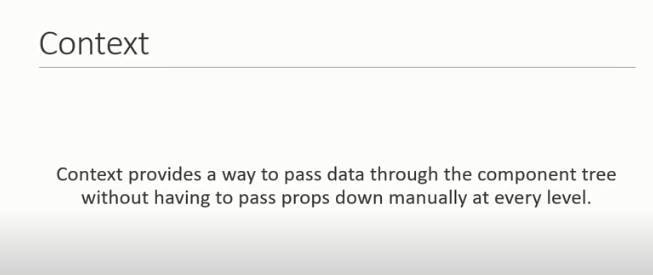
    );

}

export default DataFetching;

UseContext:





import React from 'react'

import './App.css';

import ClassCounter from './components/ClassCounter';

import HookCounter from './components/HookCounter';

import HookCounterThree from './components/HookCounterThree';

import HookCounterTwo from './components/HookCounterTwo';

import 'bootstrap/dist/css/bootstrap.min.css';

import HookCounterArr from './components/HookCounterArr';

import ClickCounterOne from './components/ClickCounterOne';

import HookCounterEffect from './components/HookCounterEffect';

import ClassMouse from './components/ClassMouse';

import HookMouse from './components/HookMouse';

import MouseContainer from './components/MouseContainer';

import IntervalClassCounter from './components/IntervalClassCounter';

import HookIntervalCounter from './components/HookIntervalCounter';

import DataFetching from './components/DataFetching';

import ComponentC from './components/ComponentC';

export const UserContext = React.createContext()

export const ChannelContext = React.createContext()

function App() {

  return (

    <div className="App">

      <UserContext.Provider value = 'Geet'>

        <ChannelContext.Provider value = 'codeEvolution'>

          <ComponentC />

       </ChannelContext.Provider>

      </UserContext.Provider>

      {/\* <DataFetching/> \*/}

      {/\* <HookIntervalCounter />

      <IntervalClassCounter /> \*/}

      {/\* <MouseContainer/> \*/}

      {/\* <HookMouse /> \*/}

      {/\* <ClassMouse /> \*/}

      {/\* <HookCounterEffect /> \*/}

      {/\* <ClickCounterOne /> \*/}

      {/\* <HookCounterArr /> \*/}

      {/\* <HookCounterThree/> \*/}

      {/\* <HookCounterTwo /> \*/}

      {/\* <HookCounter/>

     <ClassCounter/> \*/}

    </div>

  );

}

export default App;

import React,{useContext} from 'react';

import {UserContext,ChannelContext} from '../App'

function ComponentC(props) {

    const user = useContext(UserContext)

    const channel = useContext(ChannelContext)

    return (

        <div>

            {user} - {channel}

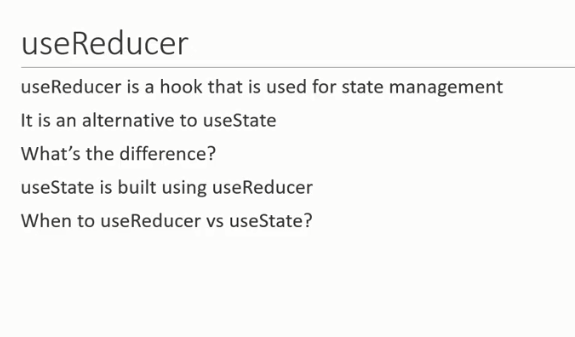
        </div> //we cn easily consume value with the help of useContext hook

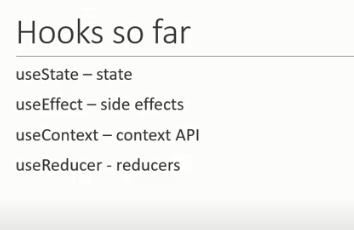
    );

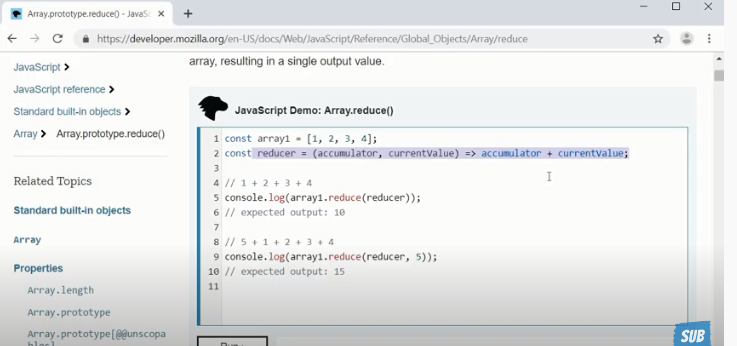
}

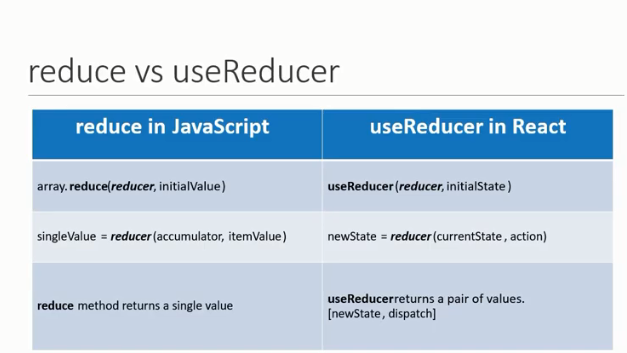
export default ComponentC;

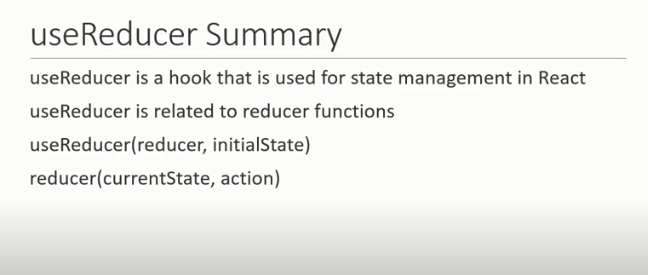
useReducer:











1. Import useReducer
2. Cal useReducer inside function
3. Initialize parameter value outside component
4. Use restructuring method and getting useReducer value that we can call in our elements.
5. import React,{useReducer} from 'react';
6. const initialState = 0;//abv comp define both parameters values
7. const reducer = (state,action) => {//reducer fn aways return one value adn we need to pass 2 paramater first is current state adn second is action
8. switch(action){
9. case 'increament' :
10. return state + 1
11. case 'decreament' :
12. return state - 1
13. case 'reset' :
14. return initialState
15. default :
16. return state
17. }
18. }
19. function CounterReducerSimple(props) {
20. const[count,dispatch] =  useReducer(reducer,initialState)//pass two parameter first is action and second we need to initialize number value
21. //as per abv using desctructing we hv store state vallue in count variable and action fn in dispatch variable
22. return (
23. <div>
24. <h3>Count : {count}</h3>
25. <button onClick = {() => dispatch('increament')}>Increament</button>&nbsp;
26. <button onClick = {() => dispatch('decreament')}>Decreament</button>&nbsp;
27. <button onClick = {() => dispatch('reset')}>Reset</button>
28. </div>
29. );
30. }
31. export default CounterReducerSimple;

import React,{useReducer} from 'react';

const initialState = {

    firstCounter:0,

    secondCounter:10

};//Add object insteed of numeric

//As we hv 2 diff propertyin state obect so we ned to mergeit for expected output so we cn use spread operator for this like ...state as below

const reducer = (state,action) => {

    switch(action.type){

        case 'increament' :

            return {...state,firstCounter:state.firstCounter + action.value}

        case 'decreament' :

            return {...state,firstCounter:state.firstCounter - action.value}

        case 'increament2' :

            return {...state,secondCounter:state.secondCounter + action.value}

        case 'decreament2' :

            return {...state,secondCounter:state.secondCounter - action.value}

        case 'reset' :

            return initialState

        default :

            return state

    }

}

function CounterReducerComplex(props) {

   const[count,dispatch] =  useReducer(reducer,initialState)//pass two parameter first is action and second we need to initialize number value

   //as per abv using desctructing we hv store state vallue in count variable and action fn in dispatch variable

   return (

        <div>

            <h3>Count 1: {count.firstCounter}</h3>

            <h3>Count 2 : {count.secondCounter}</h3>

            <button onClick = {() => dispatch({type:'increament',value:1})}>Increament</button>&nbsp;

            <button onClick = {() => dispatch({type:'decreament',value:1})}>Decreament</button>&nbsp;

            <button onClick = {() => dispatch({type:'increament',value:5})}>IncreamentBy5</button>&nbsp;

            <button onClick = {() => dispatch({type:'decreament',value:5})}>DecreamentBy5</button>&nbsp;

            <button onClick = {() => dispatch({type:'increament2',value:2})}>IncreamentSec2</button>&nbsp;

            <button onClick = {() => dispatch({type:'decreament2',value:2})}>DecreamentSec2</button>&nbsp;

            <button onClick = {() => dispatch({type:'reset'})}>Reset</button>

        </div>

    );

}

export default CounterReducerComplex;

import React,{useReducer} from 'react';

const initialState = 0;//abv comp define both parameters values

const reducer = (state,action) => {//reducer fn aways return one value adn we need to pass 2 paramater first is current state adn second is action

    switch(action.type){

        case 'increament' :

            return state + action.value

        case 'decreament' :

            return state - action.value

        case 'reset' :

            return initialState

        default :

            return state

    }

}

function CounterReducerMultiple(props) {

   const[count,dispatch] =  useReducer(reducer,initialState)

   const[countTwo,dispatchTwo] =  useReducer(reducer,initialState)//Use numitiple useReducer rather than use object adn initiliazetwoparameterin it and merge state using spread fn so try to use multiple reduce than single n complex

   return (

        <div>

            <h3>Count : {count}</h3>

            <h3>Count Two: {countTwo}</h3>

            <button onClick = {() => dispatch({type:'increament',value:1})}>Increament</button>&nbsp;

            <button onClick = {() => dispatch({type:'decreament',value:1})}>Decreament</button>&nbsp;

            <button onClick = {() => dispatchTwo({type:'increament',value:2})}>IncreamentTwo</button>&nbsp;

            <button onClick = {() => dispatchTwo({type:'decreament',value:2})}>DecreamentTwo</button>&nbsp;

            <button onClick = {() => dispatch({type:'reset'})}>Reset</button>&nbsp;

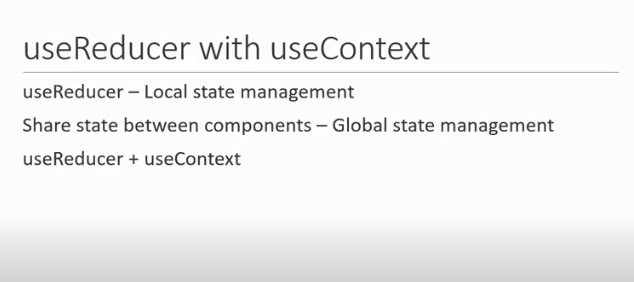
            <button onClick = {() => dispatchTwo({type:'reset'})}>ResetTwo</button>

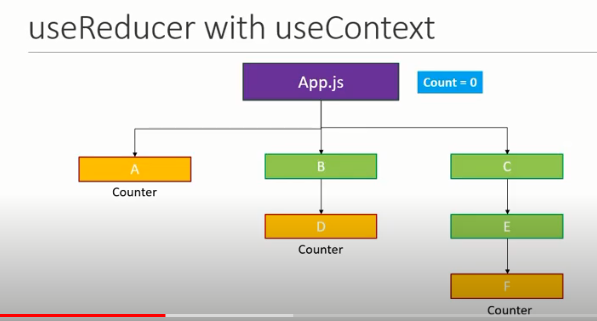
        </div>

    );

}

export default CounterReducerMultiple;





import React,{useReducer} from 'react'

import './App.css';

import ClassCounter from './components/ClassCounter';

import HookCounter from './components/HookCounter';

import HookCounterThree from './components/HookCounterThree';

import HookCounterTwo from './components/HookCounterTwo';

import 'bootstrap/dist/css/bootstrap.min.css';

import HookCounterArr from './components/HookCounterArr';

import ClickCounterOne from './components/ClickCounterOne';

import HookCounterEffect from './components/HookCounterEffect';

import ClassMouse from './components/ClassMouse';

import HookMouse from './components/HookMouse';

import MouseContainer from './components/MouseContainer';

import IntervalClassCounter from './components/IntervalClassCounter';

import HookIntervalCounter from './components/HookIntervalCounter';

import DataFetching from './components/DataFetching';

import ComponentC from './components/ComponentC';

import CounterReducerSimple from './components/CounterReducerSimple';

import CounterReducerComplex from './components/CounterReducerComplex';

import CounterReducerMultiple from './components/CounterReducerMultiple';

import ComponentUseReducerContextA from './components/ComponentUseReducerContextA';

export const UserContext = React.createContext()

export const ChannelContext = React.createContext()

export const CountContext = React.createContext()

const initialState = 0;//abv comp define both parameters values

const reducer = (state,action) => {//reducer fn aways return one value adn we need to pass 2 paramater first is current state adn second is action

    switch(action){

        case 'increament' :

            return state + 1

        case 'decreament' :

            return state - 1

        case 'reset' :

            return initialState

        default :

            return state

    }

}

function App() {

  const[count,dispatch] =  useReducer(reducer,initialState)

  return (

      <div className="App">

        <CountContext.Provider

          value = {{countState: count, countDispatch: dispatch }}

        >

          <h2>Count - {count}</h2>

          {/\* <button onClick = {() => dispatch('increament')}>Inc</button> \*/}

          <ComponentUseReducerContextA />

          </CountContext.Provider>

        {/\* <CounterReducerMultiple /> \*/}

        {/\* <CounterReducerComplex /> \*/}

        {/\* <CounterReducerSimple/> \*/}

        {/\* <UserContext.Provider value = 'Geet'>

          <ChannelContext.Provider value = 'codeEvolution'>

            <ComponentC />

        </ChannelContext.Provider>

        </UserContext.Provider> \*/}

        {/\* <DataFetching/> \*/}

        {/\* <HookIntervalCounter />

        <IntervalClassCounter /> \*/}

        {/\* <MouseContainer/> \*/}

        {/\* <HookMouse /> \*/}

        {/\* <ClassMouse /> \*/}

        {/\* <HookCounterEffect /> \*/}

        {/\* <ClickCounterOne /> \*/}

        {/\* <HookCounterArr /> \*/}

        {/\* <HookCounterThree/> \*/}

        {/\* <HookCounterTwo /> \*/}

        {/\* <HookCounter/>

      <ClassCounter/> \*/}

      </div>

  );

}

export default App;

import React,{useContext} from 'react';

import { CountContext } from '../App';

import ComponentUseReducerContextB from './ComponentUseReducerContextB';

function ComponentUseReducerContextA() {

    const countContext = useContext(CountContext)

    return (

        <div>

            <div>

                <h5>Component A - Count : {countContext.countState}</h5>

                <button onClick = {() => countContext.countDispatch('increament')}>Increament</button>&nbsp;

                <button onClick = {() => countContext.countDispatch('decreament')}>Decreament</button>&nbsp;

                <button onClick = {() => countContext.countDispatch('reset')}>Reset</button>

            </div>

            <ComponentUseReducerContextB />

        </div>

    );

}

export default ComponentUseReducerContextA;

import React,{useContext} from 'react';

import { CountContext } from '../App';

function ComponentUseReducerContextB(props) {

    const countContext = useContext(CountContext)

    return (

        <div>

           <h5>Component B - Count : {countContext.countState}</h5>

                <button onClick = {() => countContext.countDispatch('increament')}>Increament</button>&nbsp;

                <button onClick = {() => countContext.countDispatch('decreament')}>Decreament</button>&nbsp;

                <button onClick = {() => countContext.countDispatch('reset')}>Reset</button>

        </div>

    );

}

export default ComponentUseReducerContextB;

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

import axios from 'axios'

function FetchingDataWidStateEffect(props) {

    const [loading,setLoading] = useState(true)

    const [posts,setPosts] = useState({})

    const [error,setError] = useState('')

    useEffect (() => {

        axios.get('https://jsonplaceholder.typicode.com/posts/1')

        .then(response => {

            setLoading(false)

            setPosts(response.data)

            setError('')

        })

        .catch(error => {

            setLoading(false)

            setPosts({})

            setError('Something went wrong !')

        })

    })

    return (

        <div>

            {loading ? 'Loading...' : posts.title}

            {error ? error : null}

        </div>

    );

}

export default FetchingDataWidStateEffect;

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

import axios from 'axios'

const initialState = {

    loading:true,

    posts:{},

    error:''

}

const reducer = (state,action) => {

    switch(action.type){

        case 'FETCH\_SUCESS' :

            return {

                loading:false,

                posts:action.payload,

                error:''

            }

        case 'FETCH\_ERROR' :

            return {

                loading:false,

                posts:{},

                error:'Something went wrong !'

            }

        default :

        return state

    }

}

function FetchingDataWidReducer(props) {

    const [state,dispatch] = useReducer(reducer,initialState)

    useEffect (() => {

        axios.get('https://jsonplaceholder.typicode.com/posts/1')

        .then(response => {

             console.log(response.data.title);

            dispatch({type:'FETCH\_SUCESS',payload:response.data})

        })

        .catch(error => {

            dispatch({type:'FETCH\_ERROR'})

        })

    },[])

    return (

        <div>

            {state.loading ? 'Loading...': state.posts.title}

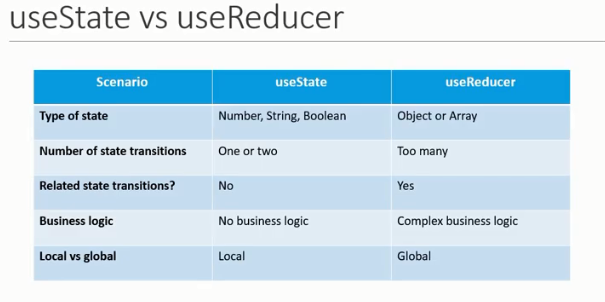
            {state.error ? state.error : null}

        </div>

    );

}

export default FetchingDataWidReducer;



CallbackHook.

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

import Button from './Button';

import Count from './Count';

import Title from './Title';

function ParentComp(props) {

    const [age,setAge] = useState(25)

    const [salary,setSalary] = useState(50000)

    const increamentAge = useCallback(() => {

        setAge(age + 1)

    },[age])

    const increamentSalary = useCallback(() => {

        setSalary(salary + 1000)

    },[salary])  //We can useEffect hook for this but only present component if we want to send any fn to another component so wen create diff fn for this n use useCallback fn for ignore call unwantes renderering fn or component if not uses

    return (

        <div>

            <Title />

            <Count text = "Age" count = {age}/>

            <Button handleClick = {increamentAge}>Increament Age</Button>

            <Count text = "Salary" count = {salary}/>

            <Button handleClick = {increamentSalary}>Increament Salary</Button>

        </div>

    );

}

export default ParentComp;

useMemo: If wewant to ignore cache value so new n use thos hook

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

function Counter(props) {

    const[counterOne,setCounterOne] = useState(0)

    const[counterTwo,setCounterTwo] = useState(0)

    const increamentCountOne = () =>{

        setCounterOne(counterOne + 1)

    }

    const increamentCountTwo = () =>{

        setCounterTwo(counterTwo + 1)

    }

    const isEven = useMemo(() =>{

        let i=0

        while(i<200000000) i++

        return counterOne % 2 === 0

    },[counterOne])//while use useMemo it will ignore cache value

    return (

        <div>

            <button onClick = {increamentCountOne}> Count - {counterOne}</button>

            &nbsp;

            <span>{isEven  ? 'Even' : 'Odd'}</span><br/><br/>

            <button onClick = {increamentCountTwo}> Count - {counterTwo}</button>

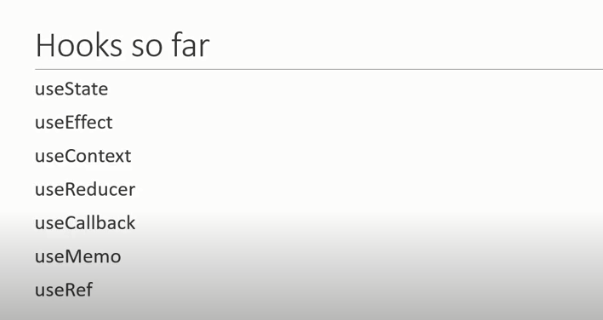
        </div>

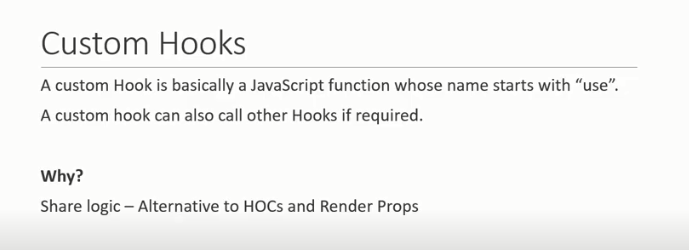
    );

}

export default Counter;

useRef-hook – refer my doc attached in git





Using custom hook we cn reuse code with custom hook attached example in react-hook code