HOOKS :- Hooks aims to solve all of these by enabling you to write functional components that have access to features like state, context, lifecycle methods , ref , etc .. without writing the class component.

1. **USESTATE () HOOK** :- state are an essential part of react . They allows us to declare state variable that hold data will be used in our app. With class components state is usually defined like this .

Class Example extends React. Component {

Constructor(props){

Super(props);

this. state = {

count : 0

};

}

Before hooks state was usually only used in a class component by as mentioned above , **Hooks allows us to add to state to functional component. Update the value of the next render. React is tracking the value of the variable and it will pass in the new value when it re-render this component.** The most important one is that you must only hooks at the top level of your function .

import React , {useState} from "react";

const **Index** = ()=>{

const [count, **setcount**]= **useState**(0);

const **incremntCount** = () =>{**setCount**(count+1);

*// setCount((prevCount)=>prevCount+1)*};

const **decrementCount** = () =>{**setCount**((prevCount)=>prevCount-1);

**setCount**((prevCount)=>prevCount-1)};

return (<div>

<button *onClick*={**decrementCount**}>-</button>

<span> count : {count}</span>

<button *onClick*={**incremntCount**}>+</button>

</div>);

};

export default **Index**;

ex-2

import React,{useState} from 'react';

let con = "Lorem, ipsum dolor sit amet consectetur adipisicing elit. Veritatis suscipit ratione blanditiis corrupti temporibus mollitia? Culpa doloremque modi nostrum denim eaque inventore, perspiciatis nates Quam voluptates option aut send voluptates magnum quo name accusation solute voluptates superiors consequent nescient doloremque parador autem in eyes? Nobi’s mayors assumed puro hic"

const **Index** = () => {

const [showdata, **setData**] = **useState**(false)

const **handleChange** = ()=>**setData**(!showdata);

return (<div>

<button *onClick*={**handleChange**}>{

showdata ? "hidden" : "show"}</button>

{showdata  ? <div>{con}</div> : <h1>Hello</h1>}

</div>

)

}

export default **Index**;

ex-3

import React,{useState} from "react";

const **Index** =  () =>{

const [firstName, **setfirstName**] = **useState**("");

const [email,**setemail**] =**useState**("");

const [pass,**setpass**] = **useState**("");

*// const changefirst=(e)=>{setfirstname(e.target.value)};*

*// const changeemail =(e)=>{setemail(e.target.value)};*

*// const changepass =(e) =>{setpass(e.target.value)};*

const **handleSubmit** = (e)=>{

e.**preventDefault**();

let userObj = {firstName: firstName,email: email, pass: pass,};

console.**log**(userObj);

};

return (<div>

<form *onSubmit*={**handleSubmit**}>

<input *type*="text" *name*="firstName" *id*="firstName" *placeholder*="enter your name" *value*={firstName} *onChange*={(e)=>**setfirstname**(e.target.value)}/><br/>

<input *type*="email" *name*="email" *id*="email"

*placeholder*="enter your email" *value*={email} *onChange*={(e)=>**setemail**(e.target.value)}/><br/>

<input *type*="password" *name*="pass" *id*="pass"

*placeholder*="enter your password" *value*={pass} *onChange*={(e)=>**setpass**(e.target.value)} /><br/>

<button *type*="submit">submit</button>

</form>

</div>

)

}

export default **Index**;

ex-4

import React ,{useState}from 'react'

const **Index** = () => {

const array = [{id:”a”,first\_Name:“Naveen",

last\_name:"chennamsetty",age:21},{id:"b",first\_name:"Nandu",

last\_name:"chennamsetty",age:20},{id:"c",first\_name:"Nari",

last\_name:"chennamsetty",age:18}];

const [data, **setData**]= **useState**(array);

const **changeFirstName** = (coming) =>{

const filterdata = data.**filter**((eachItem)=>{

return eachItem.id !== coming ;

})

**setData**(filterdata)

};

return (

<div>

<ul>

{

data.**map**((eachItem,index)=>{

const {first\_Name, last\_Name, age , id} = eachItem;

return (

<li *key*={index}>

<div>My firstName :<strong>{firstName}</strong></div>

<div>My LastName :<em>{lastName}</em></div>

<div>age:{age}</div>

<button *type*="submit" *onClick*={()=>**changeFirstName**(id)}>delete me </button>

</li>

);

})

}

</ul>

</div>

)

}

export default **Index**;

Ex-5

import React,{useState}from "react";

const **Index** = () => {

const intialObj = {first\_Name :"Naveen",Last\_name:”chennamsetty",

age:21};

const [data, **setdata**] = **useState**(intialObj);

const **changeFirstName** =()=>{

**setdata**({...data,first\_name:”Nadu",});

};

const **changeLastName** =() => {**setdata**({...data,

Last\_name:”chennamsetty");

};

return (<div>

<h1>{data.firstName}</h1>

<h2>{data.lastName}</h2>

<h3>{data.age}</h3>

<button *type*="submit" *onClick*={**changeFirstName**}>first\_Name</button><br/><button *type*="submit" *onClick*={**changeLastname**}>last\_Name</button>

</div>

)

};

export default **Index**;

Ex -6

import React, { useState } from "react";

const **Index** = () => {

const [list, setList] = **useState**([]);

const [message, setmessage] = **useState**({text: "",id: "",});

const [editingItem,**setEditingItem**]=**useState**({id:",isEditing:false});

const **changeMessage** = (e) => {

**setMessage**({...message, text: e.target.value});

};

const **handleSubmit** = (e) => {

e.**preventDefault**();

let newTodo = {text: message.text,

id: new **Date**().**getTime**().**toString**()};

**setList**([...list, newTodo]);

**setMessage**({text: "",id: ""});

};

const **handleDelete** = (id) => {

let newTodos = list.**filter**((eachItem) => {

return eachItem.id !== id;

});

**setList**(newTodos);

};

const **changeEditState** = (id) => {

**setEditingItem**({...editingItem,id: id,isEditing: true,});

let editableItem = list.**find**((eachItem) => eachItem.id === id);

**setMessage**({...message,text: editableItem.text,

id: editableItem.id});

};

const **handleEdit** = (e) => {e.**preventDefault**();

console.**log**("previous todos", list);

let newTodo = list.**map**((eachItem) => {

if (eachItem.id === editingItem.id) {

return {text: message.text, id: editingItem.id};

} else {return eachItem}

});

**setList**(newTodo);

**setMessage**({text:"",id:""});

**setEditingItem**({id: "",isEditing: false,});

};

return (

<div>

<form>

<input *type*="text" *name*="message *id*="message" *placeholder*="enter some"

*value*={message.text} *onChange*={**changeMessage**} />

{editingItem.isEditing ? (<button *onClick*={handleEdit} *type*="submit">

edit</button>):(<button *onClick*={handleSubmit} *type*="submit">add

</button>)}

</form>

<hr />{list.length === 0 && <h4>There is no items in the list</h4>

<ul>

{list.**map**((eachItem) => {

const { text, id } = eachItem;

return (<li *key*={id}>

<span>{text}</span>

<button *onClick*={() => **changeEditState**(id)}>edit</button>

<button *onClick*={() => **handleDelete**(id)}>delete</button>

</li>);

})}

</ul>

</div>

);

}

export default Index;

1. **USEEFFECT() HOOK** :- **The useEffect Hook allows you to perform side effects in the function components**. Side effects can be API calls , Updating DOM and timers. useEffect accepts two arguments .

**useEffect(<function> , <dependency> )**

**1. useEffect ( () => {**

**//It will execute every time when the component is render / re-render / mount.**

**})**

**2.useEffect ( ()=>{**

**//It will execute only when component is mounted**

**},[ ]);**

**3.Props or state values :**

**useEffect ( () => {**

**//It will execute only when dependencies changed.**

**}, [prop , state ]);**

Ex**-1**

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

const **Index** = () =>{

const [count,**setcount**] = **useState**(0)

const [pagewidth,**setpagewidth**] = **useState**(window.innerWidth);

**useEffect**(()=>{

const **resizehandle**=()=>{

**setpagewidth**(window.innerWidth)

};

window.**addEventListener**('resize',**resizehandle**);

console.**log**("Hello I am coming From UseEffect",count);

return () => {

console.**log**(" I am removing")

window.**removeEventListener**("resize",**resizehandle**)

}

});

return <div>

<h1>Learn UseEffect</h1>

<h1>{pagewidth}</h1>

<h1>{count}</h1>

<button *onClick*={(e)=>{**setcount**(count+1)}}>Click</button>

</div>

}

export default **Index**;

Ex-2 :

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

const URL = "https://jsonplaceholder.typicode.com/users";

const **Index** = ()=>{

const [usersData,**setusersData**] = **useState**([]);

const [loading,**setloading**] =**useState**(false);

const [iserror,**seterror**] =**useState**({status:false,msg:""});

const **fetchUsersData** = async (apiURL) => {

**setloading**(true);

**seterror**({status:false,msg:""});

try{

const response =await  **fetch**(apiURL);

const data=await response.**json**();

**setusersData**(data);

**setloading**(false);

**seterror**({status:false,msg:""})

if(response.status === 404){

throw new **Error**("data not found")

}

}catch(error){

**setloading**(false)

s**eterror**({status:true,msg:error.message || "something wrong!"})

}

}

**useEffect**(()=>{**fetchUsersData**(URL)},[]);

if(loading){return <div><h3>Loading....</h3></div>}

if(iserror.status){

return <div><h2*style*={{color:"red"}}>{iserror.msg}</h2></div>

}

return <div>

<h1>UseEffect example </h1>

<ul>

{usersData**.map**((eachUser)=>{

const {id,name,email} = eachUser;

return<li *key*={id}>

<div>{name}</div>

<div>{email}</div>

</li>

})

}

</ul>

</div>

}

export default **Index**;

ex-3

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

const URL = <https://www.thecocktaildb.com/api/json/v1/1/search.php?s=>

const **Index**=()=>{

const [drinksData,**setDrinksData**] = **useState**([]);

const [searchterm,**setsearchterm**] = **useState**("");

const [loading,**setloading**] = **useState**(false);

const [iserror,**seterror**] = **useState**({status:false,msg:""});

const **fetchDrink** =async (url) =>{

**setloading**(true);

**seterror**({status:false,msg:""})

try {

const response = await **fetch**(url)

const {drinks} = await response.**json**();

**setDrinksData**(drinks)

**setloading**(false);

**seterror**({status:false,msg:""})

if(!drinks){

throw new **Error**("Data not found")

}

} catch (error) {

**setloading**(false)

**seterror**({status:true,msg: error.message ||"Something went wrong"});

}

}

**useEffect**(()=>{

const correctURL = `${URL}${searchterm}`;

**fetchDrink**(correctURL)},[searchterm]);

if(iserror.status){ return <h1>{iserror.msg}</h1> }

return <div>

<form>

<input *type*="text" *name*="message" *id*="message" p*laceholder*="enter some message" *value*={searchterm} *onChange*={(e)=>**setsearchterm**(e.target.value)} />

</form>

<hr/>

{loading && <h1 *style*={{color:"red"}}>Loading....</h1>}

{!loading && <ul *className*="cocktaildata">

{drinksData.**map**((eachdrink)=>{

const {idDrink,strDrink,strDrinkThumb} =eachdrink

return <li *key*={idDrink}>

<div>

<img *src*={strDrinkThumb} *alt*={strDrink}/>

</div>

<div *className*="text">

<h1>{strDrink}</h1>

</div>

</li>

})

}

</ul>

}

</div>

};

export default **Index**;

1. USECONTEXT () :- context provides a way to pass data through component tree without having to pass props down manually at every level. React Context is a way to manage state globally. To do this without Context , we will need to pass the state as “ props “ through each nested component . This is called “prop drilling”. Context API is based on Provider and Consumers architecture .

Provider :- The Context Provider will give all the values that we need to transmit globally and any of the child-level components in the tree may require . we can use Provider to wrap the component and pass the values that the child component may expect.

Consumer :- The term “consumer” in context refers to someone who consumes the context’s values. It can retrieve the values that were passed to the same provider.

Step -1

First step is to create a Context using createContext() method that will import from react package and mark it export so that later we import that .

*import* React *from* "react";

const UserContext = React.createContext();

*export* *default* UserContext;

You can create it in separate file as well , but for now , this is I have inside myParentComponent(APP.js)

Step-2

Next we’ll use the context Provider to wrap the tree of components that need the state context and will pass the data which need to access from child components.

*import* Users *from* "./users";

*import* UserContext *from* "./context";

const UsersDashboard = ()=>{

const userName = "Naveen";

*return* (

<UserContext.Provider *value*={userName}>

<div>

<h1>Use context hook</h1>

<Users/>

</div>

</UserContext. Provider>

)

}

*export* *default* UsersDashboard;

Step-3

Suppose our innermost component ChildComponent3 needs authenticatedUserName which we have passed to our context. So for that we can use useContext hook to read the state.

*import* React,{useContext} *from* "react";

*import* UserContext *from* "./context";

const Users = ()=>{

const data = useContext(UserContext)

console.log(data)

*return* (

<div>

<h5>Users</h5>

<Details/>

</div>

)

}

*export* *default* Users;

1. USEREDUCER() :- The useReducer Hook is similar to the useState Hook. It allows for custom state logic. If you find yourself keeping track of multiple pieces of state that rely on complex logic , useReducer may be useful.

Syntax :-

useReducer (<reducer>,<initialState>)

* The reducer function contains your custom state logic and the initialState can be simple value generally will contain an object.
* The useReducer Hook returns the current state and a dispatch method.

Ex:-1

import React,{ useReducer } from "react";

const reducer = (state,action)=>{

    if(action.type === "DELETE"){

        let newFilter = state.data.filter((eachItem)=>{

            return eachItem.id !== action.payload;

        });

        return {

            ...state,

            data:newFilter,

            length:state.length-1,

        }

    }

    return state;

}

const Index = ()=>{

    const intialState = {

        data:[{id: "hadh",firstName:"Naveen",email:”[naveen@gmail.com](mailto:naveen@gmail.com)”},

            {id: "bvdsb",firstName : "Kumar", email:"Kumar@gmail.com”},

            {id: "hadbubf",firstName : "Nari",email:"Nari@gmail.com"}],

        length:3}

    const handleDelete = (id)=>{

        dispatch({type: "DELETE", payload: id})

    }

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

    return(<div>

            <h1>{state.length}</h1>

          {state.data.map((eachItem)=>{

                const {id,firstName,email} = eachItem;

                return (<div key={id}>

                        <h3>{firstName}</h3>

<h1>{email}</h1>

                        <button onClick={()=>handleDelete(id)}>DELETE</button>

                    </div>

                )

            })

          }

        </div>

    )

}

export default Index;

EX:-2

import React, { useReducer, useEffect, useState } from "react";

const reducer = (state, action) => {

  if (action.type === "UPDATE\_USERS\_DATA") {

    return {

      ...state,

      usersData: action.payload,

    }

  }

  if (action.type === "LOADING") {

    return {

      ...state,

      isLoading: action.payload,

    };

  }

  if (action.type === "DELETE\_USER") {

    const newUsers = state.usersData.filter(

      (eachUser) => eachUser.id !== action.payload

    );

    return {

      ...state,

      usersData: newUsers,

    };

  }

  if (action.type === "ONCLICK\_EDIT") {

    return {

      ...state,

      isEditing: action.payload,

    };

  }

  if (action.type === "UPDATE\_USER") {

    const newUsers = state.usersData.map((eachUser) => {

      if (eachUser.id === action.payload.id) {

        return {

          id: action.payload.id,

          name: action.payload.name,

          email: action.payload.email,

        };

      } else {

        return eachUser;

      }

    });

    return {

      ...state,

      usersData: newUsers,

    };

  }

  return state;

};

const Final = () => {

  const fetchUsersData = async (URL) => {

    dispatch({ type: "LOADING", payload: true });

    dispatch({ type: "ERROR", payload: { status: false, msg: "" } });

    try {

      const response = await fetch(URL);

      const data = await response.json();

      dispatch({ type: "UPDATE\_USERS\_DATA", payload: data });

      dispatch({ type: "LOADING", payload: false });

      dispatch({type:"ERROR",payload:{status:false, msg: ""}});

    } catch (error) {

      console.log(error);

      dispatch({ type: "LOADING", payload: false });

      dispatch({type: "ERROR",

        payload: { status: true, msg: error.message }});

    }

  };

  useEffect(() => {

fetchUsersData("https://jsonplaceholder.typicode.com/users");

  },[]);

  const initialState = {usersData: [],isLoading: false,

    isError: { status: false, msg: "" },

    isEditing: { status: false, id: "", name: "", email: "" }, };

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

  const handleDelete = (id) => {

    dispatch({ type: "DELETE\_USER", payload: id });

};

  const updateData = (id, name, email) => {

    dispatch({type: "UPDATE\_USER",payload:{id,name,email}});

    dispatch({type: "ONCLICK\_EDIT",

payload: { status: false, id: "", name: "", email: "" }});

  };

  if (state.isLoading) {

    return (<div><h3>Loading...</h3></div>)

  }

  return (<div>

<h2>Users Information</h2>

      {state.isEditing?.status && (

        <EditFormContainer

          id={state.isEditing.id}

          comingTitle={state.isEditing.name}

          comingEmail={state.isEditing.email}

          updateData={updateData}

        />

      )}

      {state.usersData.map((eachUser) => {

        const { id, name, email } = eachUser;

        return (

          <div key={id} style={{ background: "lightblue" }}>

            <h3>{name}</h3>

            <p>{email}</p>

            <button onClick={() => handleDelete(id)}>delete</button>

            <button

              onClick={() =>

                dispatch({

                  type: "ONCLICK\_EDIT",

                  payload: { status: true, id: id, name: name, email },

                })

              }

            >

              edit

            </button>

          </div>

        );

      })}

    </div>

  );

};

const EditFormContainer = ({ id, comingTitle, comingEmail, updateData }) => {

  const [title, setTitle] = useState(comingTitle || "");

  const [email, setEmail] = useState(comingEmail || "");

  return (

    <>

      <form>

        <input type="text" name="title" id="title" value={title}

          onChange={(e) => setTitle(e.target.value)} />

        <input type="email" name="email" id="email" value={email}

          onChange={(e) => setEmail(e.target.value)} />

        <button onClick={() => updateData(id, title, email)}> update data

        </button>

      </form>

    </>

  );

};

export default Final;

1. USEREF :- useRef() is a hook method which is used to create a reference to the DOM elements in the functional component . It can be used to access a DOM elements directly.

* Does not cause Re-renders :- If we tired to count how many times our application renders using useState Hook , we would be caught in an infinite loop since this Hook itself causes a re-render. useRef() only returns one item. It returns an Object called current. When we initialize useRef we set the initial value : useRef(0) . we can access the count by using count.current.
* Accessing DOM elements :- In general , we want to let handle all DOM manipulation . But there some instances where useRef can be used without causing issues. In React we can add a ref attribute to an element to access it directly in the DOM.

EX:-

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

const Index = () => {

    const [firstName,setFirstName] = useState('');

    const rendercount = useRef(0);

    useEffect(()=>{

    rendercount.current  = rendercount.current +1;

    })

  return (

    <>

    <input type='text' name="firstname" id="lastname"

    onChange={(e)=>{setFirstName(e.target.value)}} />

    <h5>Typing:{firstName}</h5>

    <h5>Component rendered {rendercount.current} times </h5>

    </>

  )

}

export default Index;

EX:-2

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

const Index= () => {

    const [firstName,setFirstName] = useState('');

    const inputDom = useRef("");

    const focus = ()=>{

        inputDom.current.focus();

    }

  return (

    <>

    <input ref={inputDom} type='text' name="firstname"

id="lastname" onChange={(e)=>{setFirstName(e.target.value)}}/>

    <h5>Typing:{firstName}</h5>

    <button onClick={focus}>Focus</button>

    </>

  )

}

export default  Index;

1. **USEMEMO()** :- The useMemo Hook returns a memoized value. The useMemo Hook only runs when one of its dependencies update. This can improve performance. The useMemo Hook can be used to keep expensive, resource intensive function from needlessly running.

EX:-

import React,{useState , useMemo ,useEffect} from "react";

const Index = ()=>{

    const [number,setNumber] = useState(0)

    const [dark,setDark] = useState(0);

    const doubleNumber = useMemo(()=>{

        return slowFunction(number)

    },[number]);

    const themeChange =useMemo(()=>{

        return{

        backgroundColor : dark ? "black" :"white",

        color:dark ? "white":"black",

        };

    },[dark]);

    useEffect(()=>{

      console.log("theme changed");

    },[themeChange]);

    return (

        <div>

            <input type="number" name="number" id="number"

value={number}

          onChange={(e)=>setNumber(Number(e.target.value))} />

            <button onClick={()=>setDark(!dark)}>Change theme</button>

            <h2 style={themeChange}>the number :{doubleNumber}</h2>

        </div>

    )

}

const slowFunction = (number)=>{

    for(let index=0;index<10000000000000000;index++){

        return number\*number

    }

}

export default Index;

1. **USECALLBACK()** :- The React usecallback Hook returns a memoized callback function. Think of memorization as caching a value so that it does not need to be recalculated. The usecallback Hook only runs when one of its dependencies update. This can improve performance. The usecallback and useMemo Hooks are similar. The main difference is that useMemo returns a memoized value and usecallback returns a memoized function.

EX:-

import React, { useState , useCallback } from "react";

import Button from "./Button";

import Title from "./Title";

import Count from "./Count";

const ParentComponent = ()=>{

    const [age,setAge] = useState(0);

    const [salary,setSalary] = useState(7000);

    const increment = useCallback(()=>{

        setAge(age+1)

    },[age])

    const incrementSalary = useCallback(()=>{

        setSalary(salary+1000)

    },[salary])

    return(

        <>

        <Title />

        <Count text={"age"} number={age}/>

        <Button clickHandler={increment} children ={"increment age"}/>

        <Count text={"salary"} number={salary}/>

        <Button

clickHandler={incrementSalary} children ={"incrementsalary"}/>

        </>

    )

}

export default ParentComponent;

import React from "react";

const Count = ({text,number})=>{

    console.log(`${text} render`)

    return (

        <h1>{text}:{number}</h1>

    )

}

export default React.memo(Count);

import React from "react";

const Button = ({children,clickHandler})=>{

    console.log(`${children} rendered`);

    return <button onClick={clickHandler}>{children}</button>

}

export default React.memo(Button);

import React from "react";

const Title=()=>{

    console.log('title render')

    return (

        <div>

            <h1>useCallback Demo && React.memo</h1>

        </div>

    )

}

export default React.memo(Title);

1. **REACT CUSTOM HOOKS** :- Hooks are reusable functions. When you have component logic that need to be used by multiple components , we can extract that logic to a custom Hook. Custom Hooks start with “use”.

Ex :-1

import React,{useState} from "react";

import usePageTitle from "./usePageTitle";

const Index = ()=>{

    const [count,setCount] = useState(0)

    usePageTitle(count)

    return (

        <div>

          <h1>PageTitleOne</h1>

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

        </div>

    )

}

export default Index;

import React,{useState} from "react";

import usePageTitle from "./usePageTitle";

const Index2 = ()=>{

    const [count,setCount] = useState(0)

    usePageTitle(count)

    return (

        <div>

          <h1>PageTitleOne</h1>

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

        </div>

    )

}

export default Index2;

import { useEffect} from "react";

function usePageTitle(count){

    useEffect(()=>{document.title=`count = ${count}`},[count])

}

export default usePageTitle;

Ex:-2

import React from "react";

import useCounter from "./useCounter";

const Index = ()=>{

    const [count,increment,decrement,reset] =useCounter(10)

    return (

        <div>

            <h1>count : {count}</h1>

            <div>

                <button onClick={increment}>increment</button>

                <button onClick={decrement}>decrement</button>

                <button onClick={reset}>reset</button>

            </div>

        </div>);

};

export default Index;

import React from "react";

import useCounter from "./useCounter";

const Index1 = ()=>{

    const [count,increment,decrement,reset]= useCounter(0)

    return (

        <div>

            <h1>count : {count}</h1>

            <div>

                <button onClick={increment}>increment</button>

                <button onClick={decrement}>decrement</button>

                <button onClick={reset}>reset</button>

            </div>

        </div>

    );

};

export default Index1;

import { useState } from "react";

function useCounter(intialValue=0){

    const [count,setCount] = useState(intialValue);

    const increment = ()=>{setCount(count+1)}

    const decrement = ()=>{setCount(count-1)}

    const reset = ()=>{setCount(0)}

    return [count,increment,decrement,reset]

}

export default useCounter;

Ex :-3

import React from "react";

import useFetch from "./useFetch";

const URL = "https://jsonplaceholder.typicode.com/users"

const FirstApi = ()=>{

    const [data,loading,isError] = useFetch(URL)

    if(loading){

        return <h3>Loading....</h3>

    }

    if(isError){

        return <h1>Something happened</h1>

    }

    return (

        <div>

            <h1>users</h1>

            <ul>

                {

                    data.map((eachUser)=>{

                        return <li key={eachUser.id}>{eachUser.username}</li>

                    })

                }

            </ul>

        </div>

    )

}

export default FirstApi;

import React from "react";

import useFetch from "./useFetch";

const URL = "https://jsonplaceholder.typicode.com/posts"

const SecondApi = ()=>{

    const [data, loading,isError] = useFetch(URL)

    if(loading){

        return <h3>Loading....</h3>

    }

    if(isError){

        return <h1>Something happend</h1>

    }

    return (

        <div>

            <h1>post</h1>

            <ul>

                {

                    data.map((eachUser)=>{

                        return <li key={eachUser.id}>{eachUser.title}</li>

                    })

                }

            </ul>

        </div>)

}

export default SecondApi;

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

function useFetch(URL){

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

    const [loading,setloading]=useState(false);

    const [isError,setisError] = useState(false);

    const makeAPIcall =async (URL)=>{

        setloading(true)

        setisError(false)

        try {

            const response = await fetch(URL);

            const data = await response.json();

            setData(data)

            setloading(false)

            setisError(false)

        } catch (error) {

            setloading(false)

            setisError(true)

        }

    }

    useEffect(()=>{

        makeAPIcall(URL)

    },[])

    return [data,loading,isError];

}

export default useFetch

1. REACT ROUTER DOM :- React Router Dom is used to build single-page applications. i.e. applications that have many pages or components but the page is never refreshed instead the content is dynamically fetched based on the URL. This process is called Routing and it is made possible with help of React Router DOM.

Step 1:- To create an application with multiple page routes , Let’s first start with the file structure.

Step 2:- import {BrowserRouter} from React-router-dom in index.js with inside root.render(<BrowserRouter><App/><BrowserRouter/>);

Step 3:- declare the Routes and Route in App.js with import all files from component inside the routers has structure has contain

      <Routes>

        <Route path="/" element={<Home/>}/>

        <Route path="/about" element={<About/>} />

        <Route path="/contact" element={<Contact/>} />

      </Routes>

Step 4:- Then we have to create a common Navbar which is present in all components and create import in App.js outside the Routes path then only it will be available to all Components.

import React from 'react';

import {Link } from "react-router-dom";

function Navbar(){

  return (

    <div>

        <nav>

        <Link to='/'>Home</Link>

        <Link to='/about'>About</Link>

        <Link to='/contact'>Contact</Link>

        </nav>

    </div>

  )

}

  export default Navbar;

Step 5:- with the help NavLink we can provide some more advantages like active state of the link and also we can provide styles when the link active state.

import React from 'react';

import {NavLink } from "react-router-dom";

function Navbar(){

  return (

    <div>

        <nav>

        <NavLink to='/'>Home</NavLink>

        <NavLink to='/about'>About</NavLink>

        <NavLink to='/contact'>Contact</NavLink>

        </nav>

    </div>

  )

}

  export default Navbar;

Step 6 :- we can add styles with the JavaScript with the help of isActive state inside the function.

import React from 'react';

import {NavLink } from "react-router-dom";

function Navbar(){

    const navlinkStyles =({isActive})=>{

      return {

        textDecoration : isActive ? "none" :"underline",

        fontWeigth: isActive ? "bold" : "normal",

        color:isActive?"black":"blue"

      };

    }

  return (

    <div>

        <nav>

        <NavLink style={navlinkStyles} to='/'>Home</NavLink>

        <NavLink style={navlinkStyles} to='/about'>About</NavLink>

        <NavLink style={navlinkStyles} to='/contact'>Contact</NavLink>

        </nav>

    </div>

  )

}

  export default Navbar;

* The useNavigate() hook is introduced in the React Router V6 to replace the useHistory() hook. API provides a usNavigate() hook which is an imperative version to perform the navigation actions with better compatibility.

import React from "react";

import {useNavigate} from "react-router-dom";

function Home(){

    const navigate =useNavigate();

    const navigateToSuccesspage=()=>{

        navigate('/success')

    }

    return (

        <div>

            <h1>Home</h1>

            <button onClick={navigateToSuccesspage}>Submit form</button>

        </div>

    )

}

export default Home;

* A router is a networking device that forwards data packets computer networks. Routers perform the traffic directing function between and on the global internet . Data sent through a network , such as a web page or emails, is in the form of data packets

import React from "react";

import Home from "./React-router-dom/Home";

import About from "./React-router-dom/About";

import Contact from "./React-router-dom/Contact";

import {Routes, Route} from "react-router-dom";

import Navbar from "./React-router-dom/Navbar";

import Success from "./React-router-dom/success";

import NotFound from "./React-router-dom/NotFound";

const App = () => {

  return (

    <div className="container">

      <Navbar/>

      <Routes>

        <Route path="/" element={<Home/>}/>

        <Route path="/about" element={<About/>} />

        <Route path="/contact" element={<Contact/>} />

        <Route path="/success" element={<Success/>} />

        <Route path="\*" element={<NotFound/>} />

      </Routes>

    </div>

  )

};

export default App;

Nested Routing

<Route path="/projects" element={<Projects/>}>

  <Route path="featured" element={<FeauterProjects/>}/>

      <Route path="new" element={<NewProjects/>} />

 </Route>

Dynamic Routing :- dynamic routing is the time at which the routing takes places . In dynamic routing, routes are initialized dynamically when the page gets rendered. This means almost everything is a component in the React Router. There are some interesting concepts in dynamic routing.

Lazy Loading :- Lazy Loading is a technique for waiting to load certain parts of a webpage-especially images- until they are needed. Instead of loading everything all at once , known as eager loading , the browser does not request certain resources until the user interacts in such a way that the resources are needed.

const LazyAbout = React.lazy(()=> import ('./React-router-dom/About'))

private routes :-Protected routes or private routes are those routes that refrain unauthorized users from penetrating into the React app’s pages. We will create and React app that will have certain pages that allows only those users who are authorized.

Fragments :- React Fragment is a feature in React that allows you to return multiple elements from a React component by allowing you to group a list of children without adding extra nodes to the DOM . To return elements from a React Component . you ‘ll need to wrap the element in a root element. Reacts Fragments serve as cleaner alternative to using unnecessary divs in our code.

Higher order Components :- a function that takes a component as an argument and returns a new component that wraps the original component. HOC are powerful feature of the React library. They allow you to reuse component logic across multiple components.