React JS:-

Usestate:-

The React useState Hook allows us to track state in a function component.

State generally refers to data or properties that need to be tracking in an application

useState is a react hook which is used to store variable state, it can store the variable state in the form of number, string, object, array, Boolean etc.,

example:-

import React, { useState } from 'react'

const App=()=>{

const [data,setData]= useState(0)

return(

<div>

<h1>Count:{data} </h1>

<button onClick={()=>setData(data+1)}>Add</button>

</div>

)

}

export default App;

useEffect:-

The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers.

syntax: useEffect(()=>{

//actual code

},[dependencies])

In the above syntax "dependencies" is optional parameters, if you want the code to be executed once make it empty.

If any value is specified in the dependency the code inside the first parameter gets executed whenever the dependency gets changed.

Example:-

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

const App = () => {

const [data, setData] = useState(0);

useEffect(() => {

setTimeout(() => {

setData((data) => data + 1);

}, 1000);

});

return (

<div>

<h1>Count:{data} </h1>

{*/\* <button>Add</button> \*/*}

</div>

);

};

export default App;

**useCallback hook:**

The React useCallback Hook returns a memoized callback function. Think of memoization as caching a value so that it does not need to be recalculated.

This allows us to isolate resource intensive functions so that they will not automatically run on every render. 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.

syntax: useCallback(()=>{

//actual code

},[dependencies]);

Example:-

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

import Task from "../task";

const App = () => {

const [data, setData] = useState(0);

const [tas, setTas] = useState([]);

const task = useCallback(() => {

setTas([...tas, "hello"]);

});

return (

<div>

<h1>Count:{data} </h1>

<button onClick={() => setData(data + 1)}>Add</button>

<br />

<Task task={task} tas={tas} />

</div>

);

};

export default App;

**useContext hook:**

useContext hook is a way to manage state globally.It can be used

together with the useState Hook to share state between deeply nested components more easily than with useState alone.

Example:- same component

import React, { useState, useContext, createContext } from "react";

const usercontext = createContext();

const App = () => {

const [data, setData] = useState("hello javascript");

return (

<div>

<usercontext.Provider value={data}>

<App2 />

</usercontext.Provider>

</div>

);

};

export default App;

function App2() {

const user = useContext(usercontext);

return (

<div>

<h1>hello</h1>

<p>{user}</p>

</div>

);

}

Example 2:- separatecomponent

import React, { useState } from "react";

import App2 from "../app2";

import UserContext from "../context";

const App = () => {

const [data, setData] = useState("hello javascript");

return (

<div>

<UserContext.Provider value={{ data, setData }}>

<App2 />

</UserContext.Provider>

<h1>hello</h1>

</div>

);

};

export default App;

context.js

import { createContext } from "react";

*// import ReactDOM from "react-dom/client";*

const UserContext = createContext();

export default UserContext;

app2

import React, { useContext } from "react";

import UserContext from "./context";

function App2() {

const { data, setData } = useContext(UserContext);

return (

<div>

<h1>{data}</h1>

</div>

);

}

export default App2;

**use ref :-**

The useRef Hook allows you to persist values between renders.

It can be used to store a mutable value that does not cause a re-render when updated.

It can be used to access a DOM element directly.

Example:-

How to fetch api and display the data in react only limited data.

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

const App = () => {

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

useEffect(() => {

fetch("https://jsonplaceholder.typicode.com/users")

.then((res) => res.json())

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

});

const filtered = () => {

let values = data.filter((e) => {

return e.id > 5;

});

setData(values);

};

return (

<div>

{data.map((e, id) => (

<p key={id}>{e.name}</p>

))}

<button onClick={filtered}>filter</button>

</div>

);

};

export default App;

how to use state in class-based components

import React, { Component } from "react";

export default class App extends Component {

constructor() {

super();

this.state = {

name: "hello"

};

}

changestate = () => {

this.setState ({

name: "hello world"

})

};

render() {

return (

<div>

<h1>{this.state.name}</h1>

<button onClick={this.changestate}>change</button>

</div>

);

}

}

Program to how to print specific element in the given based on ids in react.

import React, { useState } from "react";

const App = () => {

const [data, setData] = useState([1, 2, 3, 4]);

return (

<div>

{data.map((e, i) => {

if (i === 0) {

return <p key={i}>{e}</p>;

}

})}

</div>

);

};

export default App;

Program how to print specific elements from given objects based on ids in react.

import React, { useState } from "react";

const App = () => {

const [data, setData] = useState([

{

id: 1,

name: "ravi",

age: 20

},

{

id: 2,

name: "ravi",

age: 20

}

]);

return (

<div>

{data.map((e, i) => {

if (e.id === 2) {

return <p key={e.id}>{e.age}</p>;

}

})}

</div>

);

};

export default App;

import React, { useState } from "react";

const App = () => {

const [data, setData] = useState([

{

id: 1,

name: "ravi",

age: 20

},

{

id: 2,

name: "ram",

age: 20

}

]);

const [istrue, setIstrue] = useState(false);

const filtered = () => {

const res = data.filter((e, id) => {

return e.id === 1;

});

setData(res);

setIstrue(true);

};

return (

<div>

<button onClick={filtered}>GET</button>

{istrue &&

data.map((e) => {

return (

<>

<p>{e.name}</p>

</>

);

})}

</div>

);

};

export default App;

Program to create curd operations in reactjs.

import React, { useState } from 'react'

const App = () => {

  const [data,setData]=useState({

    name:'',

    email:''

  })

  const handlechange=(e)=>{

     const {name,value}=e.target;

     setData({

      ...data,

      [name]:value

     })

  }

  const [user,setUser]=useState([])

  const handlesubmit=(e)=>{

    e.preventDefault()

     setUser([...user,data])

     setData({name:'',email:''})

  }

  const handledelete =(id)=>{

      const res = [...user]

      res.splice(id,1)

      setUser(res)

  }

  const handleedit=(id)=>{

    const editedUser = user.find(user => user.id === id);

    setData(editedUser)

    const updatedUsers = user.filter(user => user.id !== id);

    setUser([...updatedUsers, data]);

  }

  return (

    <div>

      <h1>Form Data</h1>

      <form>

           <input type='text' value={data.name} name='name' placeholder='enter a name' onChange={handlechange} required/><br/>

           <input type='email' value={data.email} name='email' placeholder='enter a email' onChange={handlechange} required/>

           <button onClick={handlesubmit}>submit</button>

      </form>

      {user.map((e)=>{

          return <p>{e.name}-{e.email}

          <button onClick={()=>handledelete(e.id)}>delete</button>

          <button onClick={()=>handleedit(e.id)}>edit</button>

          </p>

      })}

    </div>

  )

}

export default App;

import React, { useState } from 'react';

import './style.css';

export default function App() {

  const [data, setData] = useState({

    name: '',

    email: '',

  });

  const [newdata, setNewdata] = useState([]);

  const [editindex, setEditindex] = useState(null);

  const handledel = (index) => {

    const res = [...newdata];

    res.splice(index, 1);

    setNewdata(res);

  };

  const handlechange = (e) => {

    setData({ ...data, [e.target.name]: e.target.value });

  };

  const handlesubmit = (e) => {

    e.preventDefault();

    if (editindex !== null) {

      const updateddata = [...newdata];

      updateddata[editindex] = data;

      setNewdata(updateddata);

      setEditindex(null);

    } else {

      setNewdata([...newdata, data]);

    }

    setData({ name: '', email: '' });

  };

  const handleedit = (index) => {

    const itemedit = newdata[index];

    setData({ name: itemedit.name, email: itemedit.email });

    setEditindex(index);

  };

  return (

    <div>

      <form>

        <input

          type="text"

          placeholder="Enter Name"

          name="name"

          value={data.name}

          onChange={handlechange}

        />

        <input

          type="email"

          placeholder="Enter Email"

          name="email"

          value={data.email}

          onChange={handlechange}

        />

        <button type="submit" onClick={handlesubmit}>

          submit

        </button>

      </form>

      {newdata.map((e, index) => {

        return (

          <>

            <p key={index}>

              {e.name} - {e.email}

              <button onClick={() => handledel(index)}>remove</button>

              <button onClick={() => handleedit(index)}>edit</button>

            </p>

          </>

        );

      })}

    </div>

  );

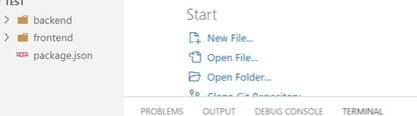
}

React life cycle methods in function



How to run with single command frontend and backend reacts and NodeJS project.

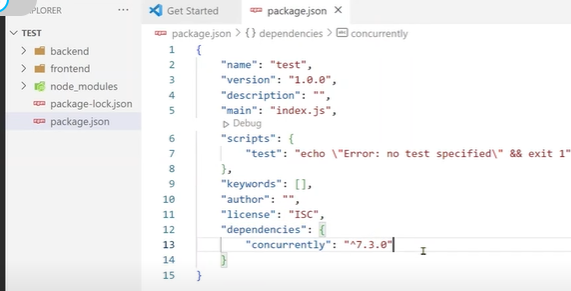
Step1: **npm init -y**



When we run this command they create package.json

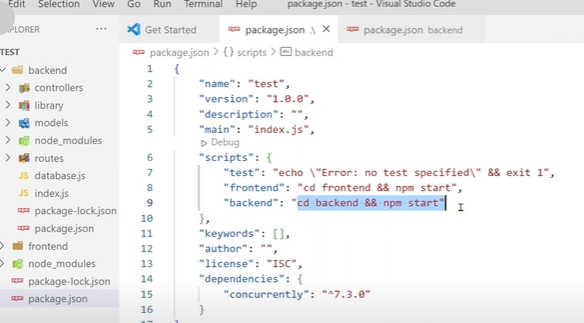
Step2: After we install one extension to run two things concurrently.

**Npm install concurrently**

****

When we install in this extension, they create node modules and package-lock. Json

Step3: after we need to add some starts to frontend and backend in package.json file to run concurrently.



Step4: At last, to create command for run the product pass as arguments in same script file.



Step5: Now we can run this project with single command.

**Npm run dev**

**Example:**

**How to install with npm install with single time to install in frontend and backend and auto start when first time npm install occurs.**

**Step 1: npm install npm-run-all --save-dev**

**After they create some extension**

**"install": "npm-run-all install:backend install:frontend", "install:backend": "cd backend && npm install --force", "install:frontend": "cd frontend && npm install --force", "postinstall": "npm start", "start": "npm-run-all --parallel start:backend start:frontend", "start:backend": "cd Backend && npm start", "start:frontend": "cd Frontend && npm start",this is put in after npm run all in package.json file**

**Step2: now npm install**

**Display repeat names from an array of objects, ensuring each name appears once in a map function, yet display multiple dates if present**.

import React from 'react';

const MyComponent = () => {

// Sample array of objects

const people = [

{ name: "Ravi", date: "2024-04-29" },

{ name: "Raju", date: "2024-04-29" },

{ name: "Ravi", date: "2024-04-30" },

{ name: "Raju", date: "2024-05-01" },

{ name: "Ramu", date: "2024-05-01" },

{ name: "shiva", date: "2024-05-01" },

{ name: "shiva", date: "2024-05-01" }

];

// Group people by name using an array

const groupedPeople = [];

people.forEach(person => {

const existingPerson = groupedPeople.find(item => item.name === person.name);

if (existingPerson) {

existingPerson.dates.push(person.date);

} else {

groupedPeople.push({ name: person.name, dates: [person.date] });

}

});

return (

<div>

{groupedPeople.map(person => (

<div key={person.name}>

<div>Name: {person.name}</div>

<div>

Dates: {person.dates.join(', ')}

</div>

</div>

))}

</div>

);

}

export default MyComponent;

**How to use React-redux-toolkit with setup in our project**

**Step1:** To this command we can setup in our react project with redux toolkit.

**npx create-react-app my-app && cd my-app && npm install @reduxjs/toolkit react-redux**

**or**

**Step1:-** To this commands also we can setup our project:

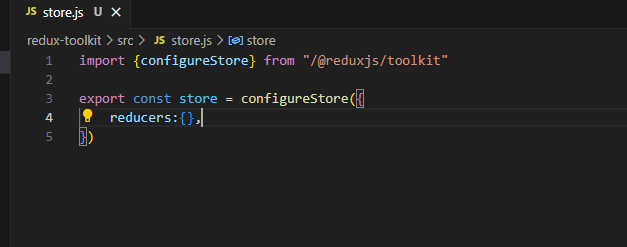
**Npx create-react-app my app**

**And**

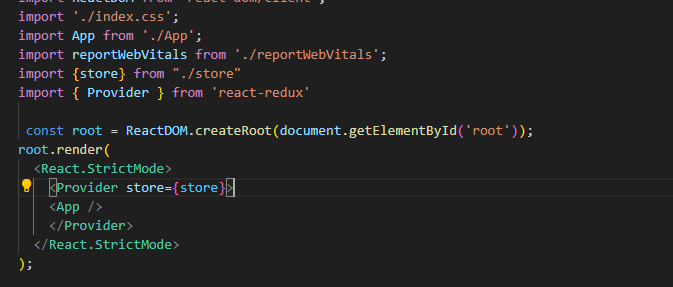
**npm install @reduxjs/toolkit react-redux**

**Step2: -** After setup we can write code here our implementation:

Here after we need to create store with store.js

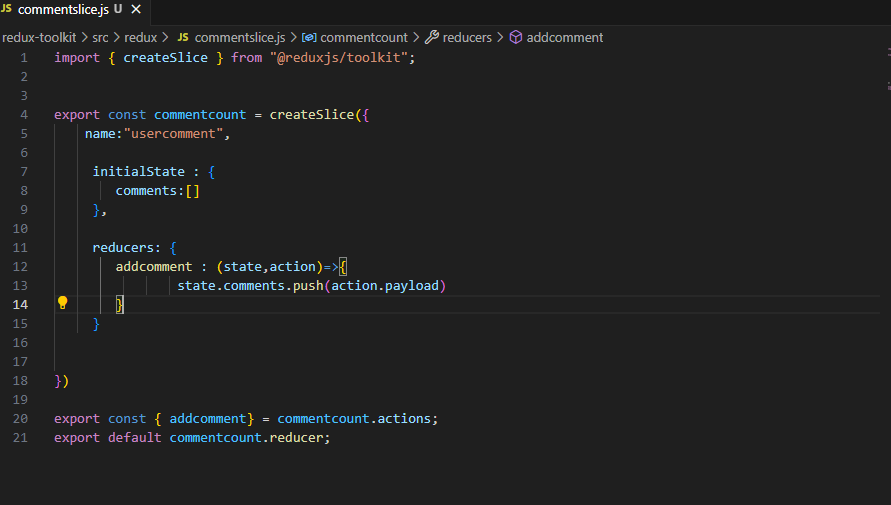


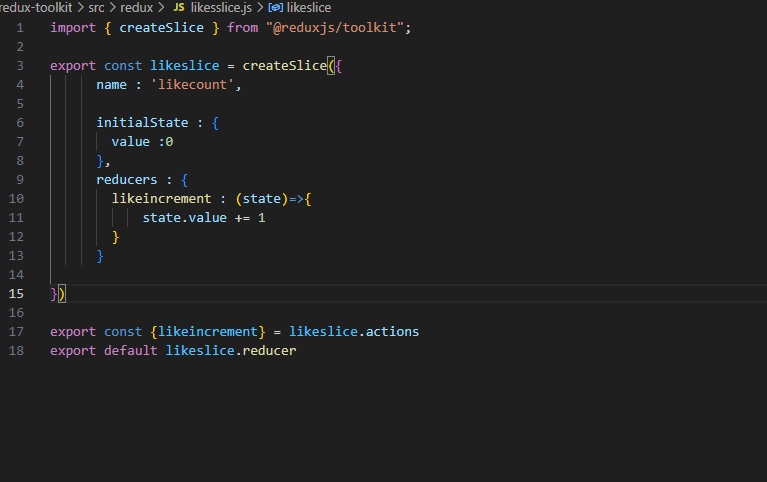
**Step3: -**Once the store is created, we can make it available to our React components by putting a React-Redux <Provider> around our application in src/index.js. Import the Redux store we just created, put a <Provider> around your <App>, and pass the store as a prop:



**Step4: -** After we provider in index file now create slices for functionalities in separate folder like redux in that folder, we can create slices with separately.

Here slices pages:





Here these are slices insistent slice and values and functionalities

Step5: - here how to display after creating slice in to use functions and values display with use dispatch and use selector with these methods we can do it.

import React, {useState} from 'react'

import { addcomment } from './redux/commentslice';

import { useDispatch } from 'react-redux';

const CommentForm = () => {

    const [userComment, setUserComment] = useState('');

     const dispatch = useDispatch()

    const commentHandler = (e)=>{

        setUserComment(e.target.value)

    }

    const submitHandler = (e)=>{

            e.preventDefault();

            console.log("New Comments: ", userComment)

            dispatch(addcomment(userComment))

            setUserComment("")

    }

  return (

    <div>

        <form className='formSection' onSubmit={submitHandler}>

            <h4>Leave your comment</h4>

            <div className="commentInput">

                <textarea value={userComment} onChange={commentHandler} id="" cols="50" rows="4"></textarea>

            </div>

            <button>Submit</button>

        </form>

    </div>

  )

}

export default CommentForm

import React from 'react'

import { useSelector } from 'react-redux'

const NewComments = () => {

  const comments = useSelector((state)=>state.usercomment.comments)

  return (

    <div className='commentSection'>

      comments:

      {

        comments.map((item,index)=>(

            <div key={index}>

              {item}

              </div>

        ))

      }

    </div>

  )

}

export default NewComments