How to deal with state in function component?

==>Earlier when function component were newly introduced state feature is not available in function component.

==>React js introduced hooks.

==>if we want to use any class component feature into function component then hooks will be used.

==>hook is a function that will provide class component feature in a function component.

==>for each class component feature,we are getting one corresponding hook(function) to implement it in a function component.

How to create state in function component?

==>every hook is following a naming convention:

==>name of hook must have a prefix i.e "use"

==>useState(),useRef(),useCallback() is a hook that will add state feature in a function component.

this.state={

score:0,team:"A",player:""

}

useState(initial-value1)//pass initial value of state. one state will be created

useState(initial-value2)

============================================================================

==>we want name/reference of the state and a function to update the state.and this things will be returned by useState function.we will receive these two things and we have to give them name.

==>if we have three state then 3 different function will be there to change the value of the state.

==>const [statename/ref , function to update the state]=useState(initial-value);

==>const [score,setScore]=useState(initial-value);

eg: const [score,setScore]=useState(0);

//read state value by using score

//update by using setScore();

eg: const [player,setPlayer]=useState("rohan");

===================================================================================

program:

import { useState } from "react";

export function FunScorecard(){

const[score,setScore]=useState(0);

const[player,setPlayer]=useState("rohan");

const incrementScore=()=>{

setScore(score+4);

}

const decrementScore=()=>{

setScore(score-1);

}

const changePlayerName=()=>{

setPlayer("suraj");

}

return(

<div>

<h1>{player}has scored{score}</h1>

<button onClick={incrementScore}>+</button>

<button onClick={decrementScore}>-</button>

<button onClick={changePlayerName}>Change Player</button>

</div>

);

}

import logo from './logo.svg';

import './App.css';

import { FunScorecard } from './FunScorecard';

function App() {

return(

<div>

<FunScorecard/>

</div>

);

}

export default App;

===============================================

LifeCycle of a react component

phases

mounting

update

unmounting

lifecycle function

1)constructor()

2)render()===>whenever state change render() will be called

//both the function belong to mounting phase.

3)componentDidMount()=>called just after first render. if we want to call api for fetching and displaying data.

this code will write in this method.

4)componentDidUpdate()==>when component has been updated .ie after the render execution when state has been updated. use for message log if the component is updated or not.

5)componentWillUnmount()==>when component will be unmounted,so before unmounting this function will be called.cleanup operations will be perform, cancel requests

========================================================================================

==>data coming from api will be stored in users key of state.

then iteration will perform on users array.

import { Component } from "react";

export class UserData extends Component

{

constructor(){

super();

this.state={

users:[]

}

}

render(){

return(

<div>

<h1>list of users1</h1>

<table border={1} cellPadding={10} cellSpacing={0}>

<thead>

<tr>

<th>id</th>

<th>Name</th>

</tr>

</thead>

<tbody>

{

this.state.users.map((u)=>{

return(

<tr>

<td>{u.id}</td>

<td>{u.name}</td>

</tr>

)

})

}

</tbody>

</table>

</div>

);

}

componentDidMount(){

fetch('https://jsonplaceholder.typicode.com/users').then((response)=>{

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

this.setState({users:data});

});

}).catch((error)=>{

console.log(error);

})

}

}

==map() will return new array by manipulating the elements of existing array.

=============================================================================================================================

Form Validations:

we can perform the following validations

1) required

2) email

3) min and max length

Eg: password should be minimum 5 characters and maximum 10 characters

Eg:

<label for = "password">Password:</label>

<input type = "password" name = "password" id = "password" pattern = ".{5,10}"

placeholder = "your password" required title = "5 to 10 characters">

Pattern for only digits:[0-9]{5,10}

Pattern for only word characters:\w{5,10}