React<16.8 use class-based components to manage states with side effect..this the find react hook to handle this.

Class based component cant use react hooks!

import classes from './User.module.css';

import { Component } from 'react';

class User extends Component {

  render(){

    return <li className={classes.user}>{this.props.name}</li>;

  }

}

// const User = (props) => {

//   return <li className={classes.user}>{props.name}</li>;

// };

export default User;

in class based component state always is an object.

And it also has to be a property named “state”! This name is not up to you.

States in class-based components:

import { Component } from "react";

import User from "./User";

import classes from "./Users.module.css";

const DUMMY\_USERS = [

  { id: "u1", name: "Max" },

  { id: "u2", name: "Manuel" },

  { id: "u3", name: "Julie" },

];

class Users extends Component {

  constructor(){

    super()

    this.state= {

      showUsers: true,

      more:"Test"

    }

   // const [showUsers, setShowUsers] = this.state(true);

  }

  toggleUsersHandler () {

    this.setState((curState) => {

      return{showUsers:!curState.showUsers}

    })

  }

  render() {

    const usersList = (

      <ul>

        {DUMMY\_USERS.map((user) => (

          <User key={user.id} name={user.name} />

        ))}

      </ul>

    );

    return (

      <div className={classes.users}>

      <button onClick={this.toggleUsersHandler.bind(this)}>

        {this.state.showUsers ? "Hide" : "Show"} Users

      </button>

      {this.state.showUsers && usersList}

    </div>

    )

  }

export default Users;

lifecycle:

component did mount():

import { Fragment, useState, useEffect, Component } from "react";

import UsersContext from "../store/users-context";

import classes from "./UserFinder.module.css";

import Users from "./Users";

class UserFinder extends Component {

    static contextType = UsersContext;

  constructor() {

    super();

    this.state = { searchTerm: "",filterUsers:[] };

  }

  componentDidMount(){

    this.setState({filterUsers:DUMMY\_USERS})

  }

  componentDidUpdate(prevsProps,prevsState){

    console.log(this.state.searchTerm)

    if(prevsState.searchTerm !==this.state.searchTerm){

        this.setState({filterUsers:DUMMY\_USERS.filter((user) => user.name.toLowerCase().includes(this.state.searchTerm.toLowerCase()))})

    }

  }

  searchChangeHandler(event){

    this.setState({searchTerm : event.target.value});

  }

  render() {

    return (

      <Fragment>

        <div className={classes.finder}>

          <input type="search" onChange={this.searchChangeHandler.bind(this)} />

        </div>

        <Users users={this.state.filterUsers} />

      </Fragment>

    );

  }

}

export default UserFinder;

using context in class-based:

we can use only one context in classbased components as follows

import { Fragment, useState, useEffect, Component } from "react";

import UsersContext from "../store/users-context";

import classes from "./UserFinder.module.css";

import Users from "./Users";

class UserFinder extends Component {

    static contextType = UsersContext;

  constructor() {

    super();

    this.state = { searchTerm: "",filterUsers:[] };

  }

  componentDidMount(){

    this.setState({filterUsers:this.context.users})

  }

  componentDidUpdate(prevsProps,prevsState){

    console.log(this.state.searchTerm)

    if(prevsState.searchTerm !==this.state.searchTerm){

        this.setState({filterUsers:this.context.users.filter((user) => user.name.toLowerCase().includes(this.state.searchTerm.toLowerCase()))})

    }

  }

  searchChangeHandler(event){

    this.setState({searchTerm : event.target.value});

  }

  render() {

    return (

      <Fragment>

        <div className={classes.finder}>

          <input type="search" onChange={this.searchChangeHandler.bind(this)} />

        </div>

        <Users users={this.state.filterUsers} />

      </Fragment>

    );

  }

}

export default UserFinder;

Error boundaries:

import classes from './User.module.css';

import { Component } from 'react';

class ErrorBountry extends Component {

    constructor(){

        super()

        this.state={hasError:false}

    }

  componentDidCatch(error){

    this.setState({hasError:true})

    alert("Error")

  }

  render(){

    if(this.state.hasError){

        return <p>Something went wrong</p>

    }

    return this.props.children;

  }

}

// const User = (props) => {

//   return <li className={classes.user}>{props.name}</li>;

// };

export default ErrorBountry;

  componentDidUpdate(){

    if(this.props.users.length == 0){

      throw new Error('No user Provided!!')

    }

  }

Error boundary only possible in class based components