```
/*
    ============== React Class Component ===================

    Clas-based Components: An Alternative To Functions

*/
// Function base Component
function Product(props) {
    return <h2>A product!</h2>
}

/*
    Components are regular JavaScritp function which return
    renderable results (typically JSX)
*/

// Class Base Component

class Product extends Component {
    render() {
        return <h2>AProduct!</h2>
    }
}


/*
    Components can also be defined as JS classes where a render() method
    defines the to-be-rendered output
*/


// ******************** convert function to to class base component ********************

// function base component *********** START ****************

import React, { useEffect } from 'react';
import Styles from './User.module.css';

const User = (props) => {

    return (
        <li clasName={Styles.user}>{props.name}</li>
    )
}
export default User;

// function base component *********** END ****************



// class base Component *********** START ********************

import Styles from '.User.module.css';

// we need to import component from react and
import React, { Component } from 'react';

// and extends that Component
// and extends that Component
class User extends Component {
    constructor() {
        // initialize somthing first
    }

    // React find the render to to display on screen by call React
    render() {
        // This render method return the JSX
        return (
            // props get the class base autometically coz we extends Component from React by (this) we can use
            <li clasName={Styles.user}>{this.props.name}</li>
        );
    }

}
// class base Component  ********************* END *********************


// ****************** Another Convert function Base to Class base Component***************

// function base component *********** START ****************

import { useState } 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' },
];

const Users = () => {
    const [showUsers, setShowUsers] = useState(true);

    const toggleUsersHandler = () => {
        setShowUsers((curState) => !curState);
    };

    const usersList = (
        <ul>
            {DUMMY_USERS.map((user) => (
                <User key={user.id} name={user.name} />
            ))}
        </ul>
    );

    return (
        <div className={classes.users}>
            <button onClick={toggleUsersHandler}>
                {showUsers ? 'Hide' : 'Show'} Users
            </button>
            {showUsers && usersList}
        </div>
    );
};

```

```
export default Users;



// function base component *********** END *****************


// class base Component  *********** START ****************
import { Component } from 'react';
import User from './User';

import classes from './Users.module.css';
import { thisStringValue } from 'es-abstract/es2019';

const DUMMY_USERS = [
    { id: 'u1', name: 'Max' },
    { id: 'u2', name: 'Manuel' },
    { id: 'u3', name: 'Julie' },
];

class Users extends Component {

    // Define ******state *********** inside the class
    // two things is needed
    // 1) Define and 2) update
    constructor() {
        super();
        // in class base component state Always object {} and only one state in class base
        // one important things is that when we update the State then the objest
        // will Marge on overwrite
        this.state = {
            showUsers: true,
            // moreState: "test",
            // nested: {},
            // data: []
            // All are working
        };
    }


    toggleUsersHandler() {
        // method define like inthis way outsite the render method
        // this.state.showUsers = false  // NOT !
        // but
        // this.setState({showUsers: false});   work it another way by class functions
        this.setState((curState) => {
            return { showUsers: !curState.showUsers }
        });
    }


    render() {

        // add userList inside the Rander Method
        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.sate.showUsers && usersList}
            </div>
        );
    }
}

// class Base Component *********** END ****************


// ================ Class-based Component Lifecycle ===================
/*
    1) Side-effects in functional Components: useEffect()
    2) Class-based Components can't use React Hooks!
*/

/*
    1) componentDidMount()
        *** Called once component mounted (was evaluated & rendered)
        *** useEffect(..., []); equvalent to without dependencies
    2) componentDidUpdate()
        **** Callded once component updated (was evaluated & rendered)
        **** useEffect(..., [dependencies]);
    3) componentWillUpdate()
        **** Called right before component is unmounted (removed from DOM)
        **** useEffect(()=> {return ()=>{...}}, []); equvalent

*/


// Functions Base *********************
DUMY_USERS = [

]
const UserFinder = () => {

    useEffect(() => {
        setFileredUsers(

            DUMY_USERS.filter((user) => user.name.includes(searchTerm))
        );
    }, [searchTerm]);


    return (
        <p>Mehedi</p>
    );
}


// Class Base *******************
```

```
import React, { Component } from 'react';

class UserFinder extends Component {
    constructor() {
        super();
        this.state = {
            filteredUsers: DUMY_USERS,
            searchTerm: '',
        }
    }

    // with out chacking previous state it gose like loop when state update
    // this function call again and again that is why we need
    componentDidUpdate(prevProps, prevState) {

        if (prevState.searchTerm !== this.state.searchTerm) {
            this.setState({
                filteredUsers: DUMMYUSERS.filter((user) =>
                    user.name.includes(searchTerm)
                )
            });
        }

    }
    searchChangerHandeler(event) {
        this.setState({ searchTerm: event.target.value }); // marge that object
    }

    render() {
        return (
            <React.Fragment>
                <div className={Styles.finder}>
                    <input type="search" onChange={this.searchChangeHandler.bind(this)} />
                </div>
                <Users users={this.state.filteredUsers} />

            </React.Fragment>

        );
    }
}


// =============== context inside the classbase component ========================

// 1) create context data file which we want to share

// ********* two way to sue context in class
/*
    1) context provider and consumer BOTH class and function use this way
    2) static contextType = userContext; like this way inside the class base component

*/

// 1 one

return (
    <UserSContext.provider value={usersContext}>
```

```
        <UserFinder />
    </UserSContext.provider>
)

return (
    <UserSContext.Consumer>
        ......................
        .................
        .........
    </UserSContext.Consumer>
);


// 2 way

class UserFinder extends Component {
    static contextType = userContext;


    // use like this
    componentDidMount() {
        this.setState({ filteredUser: this.context.useers })
    }
}


// ================= Error =====================

// this is how we can generating error by won
// when this aplication run that time if no users in this list that time it show an error
componentDidMount(){
    if (this.props.users.length === 0) {
        throw new Error("No Users provided!");
    }
}

// But we can not use error like this way...Another solution to handel the erron...=> Error Boundaries

try {
    somCodeWhichMightFail();
} catch (err) {
    // handele error
}

// it will work in javascript but not in JSX
// That's why we can use nother way...Error Boundaries


// ***************** Error Boundary *****************
import { Component } from 'react';

class ErrorBoundary extends Component {

    componentDidMount() {
        // logic here wich time error we want to handele
        super();
        this.state = {hasError : false};
    }

    componentDidCatch(error){

        // we can send error to server and analyzing the error message from here

        console.error(error);
        this.setState({hasError : true});
    }

    render() {
        if(this.state.hasError){
            return <p>Something went wrong!</p>
        }
        return this.props.children;
    }
}
export default ErrorBoundary;


// ************* which Compnent generating error that will be wrap ***********8

class App extends Component {

    render() {
        return (
            <React.Fragment>
                <p>Mehedi</p>

                <ErrorBoundary >
                // this chidl component will be generating errors wrap this...
                    <Users users={this.state.filteredUsers} />
                </ErrorBoundary>

            </React.Fragment>

        );
    }

}
```