**Experiment-9: Hooks in react.**

**Task 1: USEREDUCER**

**Passing the initializer function**

**This example passes the initializer function, so the createInitialState function only runs during initialization. It does not run when component re-renders, such as when you type into the input.**

#### 

#### Passing the initial state directly

**This example does not pass the initializer function, so the createInitialState function runs on every render, such as when you type into the input. There is no observable difference in behavior, but this code is less efficient.**

**const [state, dispatch] = useReducer(reducer, initialArg, init?)**

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

**function reducer(state, action) {**

**switch (action.type) {**

**case 'incremented\_age': {**

**return {**

**name: state.name,**

**age: state.age + 1**

**};**

**}**

**case 'decremented\_age': {**

**return {**

**name: state.name,**

**age: state.age - 1**

**};**

**}**

**case 'changed\_name': {**

**return {**

**name: action.nextName,**

**age: state.age**

**};**

**}**

**}**

**throw Error('Unknown action: ' + action.type);**

**}**

**export default function Form() {**

**const [user, dispatch] = useReducer(reducer, { name: 'Krishna', age: 30 });**

**function handleButtonClicki() {**

**dispatch({ type: 'incremented\_age' });**

**}**

**function handleButtonClickd() {**

**dispatch({ type: 'decremented\_age' });**

**}**

**function handleInputChange(e) {**

**dispatch({**

**type: 'changed\_name',**

**nextName: e.target.value**

**});**

**}**

**return (**

**<div>**

**<input**

**value={user.name}**

**onChange={handleInputChange}**

**/>**

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

**Increment age**

**</button>**

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

**decrement age**

**</button>**

**<p>Hello, {user.name}. You are {user.age}.</p>**

**</div>**

**);**

**}**

**Task 2: USEMEMO**

**UseMemo(compute,dependencies)**

**import { useState, useMemo } from 'react';**

**export default function CalculateFactorial() {**

**const [number, setNumber] = useState(1);**

**const [counter, setCounter] = useState(0);**

**const factorial = useMemo(() => factorialOf(number), [number]);**

**const onChange = event => {**

**setNumber(Number(event.target.value));**

**};**

**const counterHander = () => {**

**setCounter(counter + 1);**

**};**

**return (**

**<div>**

**Factorial of**

**<input type="number" value={number} onChange={onChange} />**

**is {factorial}**

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

**Counter**

**</button>**

**<div>Counter : {counter}</div>**

**</div>**

**);**

**}**

**function factorialOf(n) {**

**console.log('factorialOf(n) called!');**

**return n <= 0 ? 1 : n \* factorialOf(n – 1**

**);**

**}**

**Task 3: CUSTOM HOOKS**

**Toggles value of a component from true to false.**

useToggle.js

**import { useState } from "react"**

**export default function useToggle(defaultValue) {**

**const [value, setValue] = useState(defaultValue)**

**function toggleValue(value) {**

**setValue(currentValue =>**

**typeof value === "boolean" ? value : !currentValue**

**)**

**}**

**return [value, toggleValue]**

**}**

App.js

**import useToggle from "./useToggle"**

**export default function ToggleComponent() {**

**const [value, toggleValue] = useToggle(false)**

**return (**

**<div>**

**<div>{value.toString()}</div>**

**<button onClick={toggleValue}>Toggle</button>**

**<button onClick={() => toggleValue(true)}>Make True</button>**

**<button onClick={() => toggleValue(false)}>Make False</button>**

**</div> )}**