Q) Discuss the use of the useState hook in managing form state in React.  
Ans)

In React, useState is a special function that lets you add state to functional components. It provides a way to declare and manage state variables directly within a function component. It should be noted that one use of useState() can only be used to declare one state variable. It was introduced in version 16.8.

**Importing the useState Hook**

To import the useState hook, write the following code at the top level of your component

import { useState } from "react";

**Structure of React useState hook**

This hook takes some initial state and returns two value. The first value contains the state and the second value is a function that updates the state. The value passed in useState will be treated as the default value.

Syntax:

const [var, setVar] = useState(0);

**Internal working of useState hook**

* useState() creates a new cell in the functional component’s memory object.
* New state values are stored in this cell during renders.
* The stack pointer points to the latest cell after each render.
* Deliberate user refresh triggers stack dump and fresh allocation.
* The memory cell preserves state between renders, ensuring persistence.

**Handling input forms with useState:**

Handling input forms with useState in React involves creating state variables to store the values of input fields and updating them as the user interacts with the form.

* **State Variables**: Define state variables to hold the values of input fields. Each input field typically has its own state variable.
* **Binding Input Values**: Bind the value of each input field to its corresponding state variable. This ensures that the input field displays the current value stored in the state.
* **Event Handlers**: Create event handler functions to update the state variables as the user enters or modifies input. These functions typically listen for events like onChange for text inputs or onClick for buttons.
* **Updating State**: When the user interacts with the input fields, the event handlers update the state variables using setState, triggering a re-render to reflect the changes in the UI.
* **Submitting the Form**: When the user submits the form, you can access the values stored in the state variables to perform further actions, such as validation or sending data to a server.

Below is an example of handling input forms with useState():  
  
import React, {useState} from 'react';

import './App.css';

const FormExample = () => {

const [name, setName] = useState('');

const [email, setEmail] = useState('');

const handleSubmit = (event) => {

// Prevent default form submission

event.preventDefault();

console.log('Name:', name);

console.log('Email:', email);

};

return (

<div className="form-container">

<h2>Input Form</h2>

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Name:</label>

<input

type="text"

value={name}

onChange={(e)=>setName(e.target.value) }

/>

</div>

<div className="form-group">

<label>Email:</label>

<input

type="email"

value={email}

onChange={(e)=>setEmail(e.target.value)}

/>

</div>

<button type="submit">Submit</button>

</form>

</div>

);

};

export default FormExample;

**Output:**(This is a GIF displaying the process)

