Roll No :- 2105048
Name:- Manav Jain
Date:-08/09/2023

Assignment No-7b

Aim:-

WAP to implement concept of React hooks

LO Mapped:- LO5

Theory:-

React Hooks are a feature introduced in React 16.8 that allow you to use state and other React features in functional components. Before Hooks, state and other React features were only available in class components.

Some of the most commonly used React Hooks are:

useState: useState allows functional components to manage state. It returns an array with the current state value and a function to update it. Here's an example:

Eg:-
```
import React, { useState } from 'react'
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

useEffect: useEffect allows you to perform side effects in your components, such as data fetching, manually changing the DOM, or setting up subscriptions. It takes two arguments: a function to run the side effect and an array of dependencies that determine when the effect should run.

Eg:-
```
import React, { useState, useEffect } from 'react';
function Example() {
  const [count, setCount] = useState(0);
  useEffect(() => {
    document.title = `Count: ${count}`;
  }, [count]);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

useContext: useContext allows you to access the context provided by a Context.Provider higher up in the component tree. It is particularly useful for managing global state and avoiding prop drilling.
Eg:-
```
import React, { useContext } from 'react';
const MyContext = React.createContext();
function MyComponent() {
  const contextValue = useContext(MyContext);
  return <p>Value from context: {contextValue}</p>;
}
```

useReducer: useReducer is a hook for managing complex state logic. It is similar to useState, but it uses a reducer function to determine the new state based on the previous state and an action. It's often used when state transitions are more complex than simple updates.
Eg:-
```
import React, { useReducer } from 'react';
function counterReducer(state, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
}

function Counter() {
  const [state, dispatch] = useReducer(counterReducer, { count: 0 });
  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'INCREMENT' })}>Increment</button>
      <button onClick={() => dispatch({ type: 'DECREMENT' })}>Decrement</button>
    </div>
  );
}
```

React provides several other built-in hooks, and you can also create your custom hooks to reuse stateful logic across different components. Hooks make it easier to manage state and side effects in functional components, making React development more flexible and expressive.

Output:-

1)Code for app.js :-

```
import React from 'react';
import NameForm from './NameForm';

function App() {
  return (
    <div className="App">
      <NameForm />
    </div>
  );
}
export default App;
```

2)Code for index.js :-

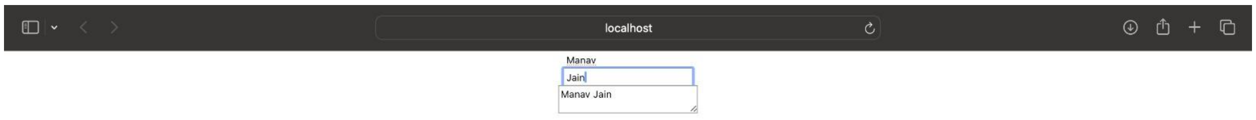
```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
ReactDOM.render(
  <React.StrictMode>
```

```
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

3)Code for NameForm.js

```
import React, { useState } from 'react';
function NameForm() {
  const [firstName, setFirstName] = useState('FirstName_initial');
  const [lastName, setLastName] = useState('Lastname_initial');
  const handleFirstNameChange = (e) => {
    setFirstName(e.target.value);
  };
  const handleLastNameChange = (e) => {
    setLastName(e.target.value);
  };
  return (
    <div>
      <h2>Name Form</h2>
      <div>
        <label htmlFor="firstName">First Name:</label>
        <input
          type="text"
          id="firstName"
          value={firstName}
          onChange={handleFirstNameChange}
        />
      </div>
      <div>
        <label htmlFor="lastName">Last Name:</label>
        <input
          type="text"
          id="lastName"
          value={lastName}
          onChange={handleLastNameChange}
        />
      </div>
      <div>
        <p>First Name: {firstName}</p>
        <p>Last Name: {lastName}</p>
      </div>
    </div>
  );
}
export default NameForm;
```

4)Output screenshots



Manav

Jain

Manav Jain

References :-

1)https://legacy.reactjs.org/docs/hooks-intro.html
2)https://www.w3schools.com/react/react_hooks.asp

Conclusion :-

Learnt about react hooks , their usage and also explored more about most commonly used react hooks like usestate , useeffect and written a simple program to demonstrate usage of react hooks