EXPERIMENT NO – 03

Aim

To create a simple React application utilizing components, props, state, and lifecycle methods. Implement basic interactivity by handling user input and managing component lifecycle events.

Theory

React is a JavaScript library for building user interfaces. It allows developers to create reusable components, manage dynamic state, and handle events efficiently.

1. Setting Up a React Project

Before coding, we need to set up a React project.

Steps to Set Up:

* 1. Install Node.js (if not installed) from nodejs.org.
  2. Create a new React app using the command:
  3. npx create-react-app my-app
  4. Navigate into the project folder:
  5. cd my-app
  6. Start the development server:
  7. npm start

This will launch the app in your browser.

1. Understanding Components and Props

Components in React

A component is a reusable UI element in React. It can be either a functional or class component.

Example: Functional Component

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

Here, Greeting is a functional component that receives props and displays the passed name.

Props (Properties)

Props allow data to be passed from a parent component to a child component. They are immutable (read-only).

Example of Passing Props:

<Greeting name="John" />

Output:

Hello, John!

3. Understanding State and Lifecycle

State in React

State is a built-in object used to store dynamic data within a component. Unlike props, state is mutable and can change over time.

Example of State in a Functional Component:

import { useState } from 'react'; function Counter() { const [count, setCount] = useState(0);

return (

<div>

<p>Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Increase</button> </div>

);

}

Explanation:

* useState(0) initializes a state variable count with 0.
* setCount(count + 1) updates the state when the button is clicked.

Lifecycle in React

React components go through different lifecycle phases:

1. Mounting – When the component is created and inserted into the DOM.
2. Updating – When state or props change, causing a re-render.
3. Unmounting – When the component is removed from the DOM.

In functional components, the useEffect hook is used to manage side effects and lifecycle behavior.

Example of useEffect Hook:

import { useEffect, useState } from 'react';

function Timer() { const [time, setTime] = useState(0);

useEffect(() => { const interval = setInterval(() => { setTime((prevTime) => prevTime + 1);

}, 1000); return () => clearInterval(interval);

}, []);

return <p>Time: {time} seconds</p>;

}

Explanation:

* useEffect runs after the component mounts.
* setInterval updates the time state every second.
* The cleanup function clearInterval(interval) stops the timer when the component unmounts.

4. Handling Events in React

React handles events similarly to standard DOM events, but uses camelCase naming (e.g., onClick instead of onclick). Example: Handling Input Change

import { useState } from 'react';

function InputHandler() { const [text, setText] = useState(""); return (

<div>

<input type="text" onChange={(e) => setText(e.target.value)} />

<p>You typed: {text}</p>

</div>

);

}

Explanation:

* onChange captures user input and updates the state text.
* The updated text value is displayed dynamically.

Summary of Key Concepts

Concept Description

Components Reusable UI elements in React.

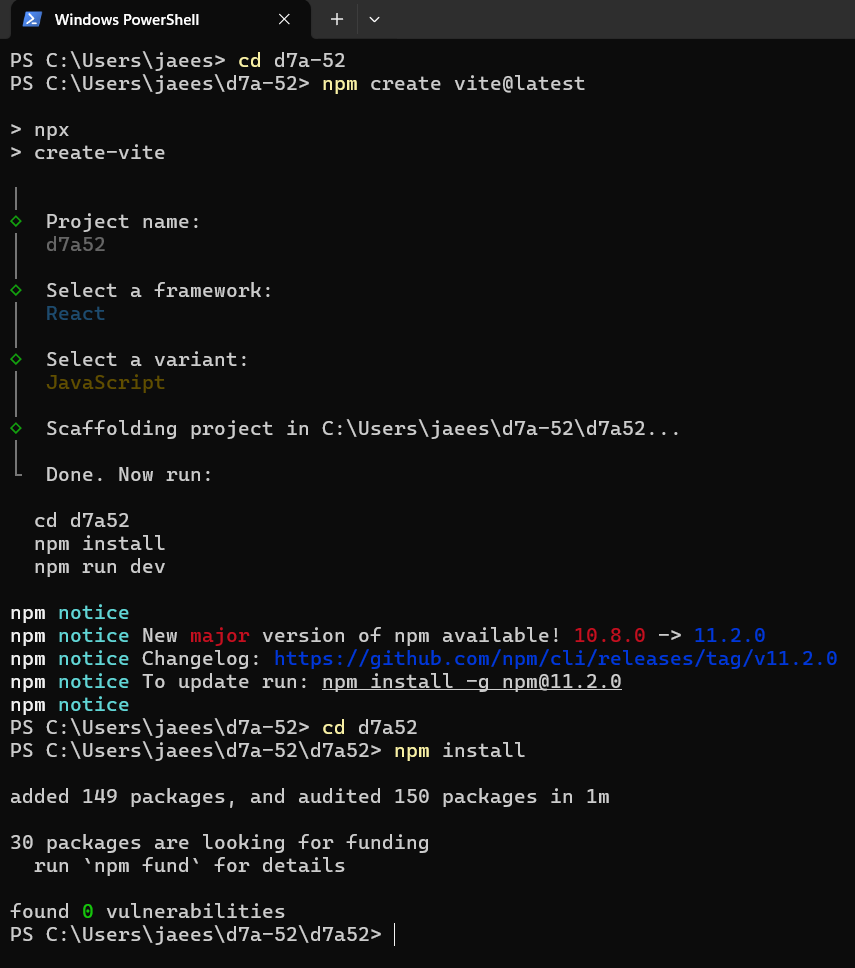
Props Pass data from parent to child (read-only).

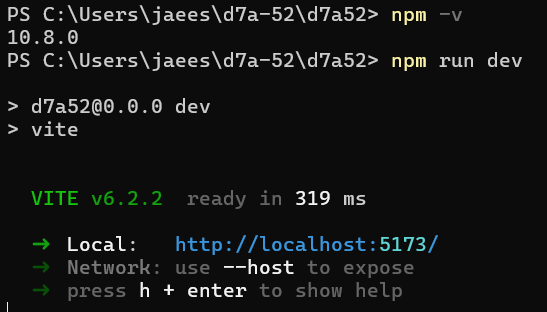
State Stores dynamic data inside a component.

Lifecycle Phases of a component (mounting, updating, unmounting).

Event Handling Capturing user interactions like clicks and input changes.

SETTING UP REACT APPLICATION





CODE:

BLOG POST

App.jsx

import React, { useState, useEffect } from 'react';

import BlogPost from './components/BlogPost';

import BlogForm from './components/BlogForm';

function App() {

const [posts, setPosts] = useState([]);

useEffect(() => {

const savedPosts = localStorage.getItem('blog-posts');

if (savedPosts) setPosts(JSON.parse(savedPosts));

}, []);

useEffect(() => {

localStorage.setItem('blog-posts', JSON.stringify(posts));

}, [posts]);

const handleNewPost = (title, author, content) => {

setPosts([{ id: Date.now(), title, author, content, date: new Date().toLocaleDateString() }, ...posts]);

};

const handleDeletePost = (id) => {

setPosts(posts.filter(post => post.id !== id));

};

return (

<div className="app">

<header className="header"><h1>My Simple Blog</h1></header>

<BlogForm onSubmit={handleNewPost} />

<div className="blog-list">

{posts.map(post => (

<BlogPost key={post.id} {...post} onDelete={() => handleDeletePost(post.id)} />

))}

</div>

</div>

);

}

export default App;

BLOGFORM.JSX

import React, { useState } from 'react';

function BlogForm({ onSubmit }) {

const [title, setTitle] = useState('');

const [author, setAuthor] = useState('');

const [content, setContent] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

if (title && author && content) {

onSubmit(title, author, content);

setTitle('');

setAuthor('');

setContent('');

}

};

return (

<form className="blog-form" onSubmit={handleSubmit}>

<div className="form-group">

<label>Title:</label>

<input type="text" value={title} onChange={(e) => setTitle(e.target.value)} placeholder="Enter post title" />

</div>

<div className="form-group">

<label>Author:</label>

<input type="text" value={author} onChange={(e) => setAuthor(e.target.value)} placeholder="Enter author name" />

</div>

<div className="form-group">

<label>Content:</label>

<textarea value={content} onChange={(e) => setContent(e.target.value)} placeholder="Write your blog post here" />

</div>

<button type="submit" className="submit-button">Create Post</button>

</form>

);

}

export default BlogForm;

BlogPost.jsx

import React from 'react';

function BlogPost({ title, author, content, date, onDelete }) {

return (

<div className="blog-post">

<button className="delete-button" onClick={onDelete}>Delete</button>

<h2>{title}</h2>

<div className="author">By {author}</div>

<div className="date">{date}</div>

<div className="content">{content}</div>

</div>

);

}

* export default BlogPost;

ToDoItem.jsx

import React from 'react';

function TodoItem({ text, completed, onToggle, onDelete }) {

return (

<li className={`todo-item ${completed ? 'completed' : ''}`}>

<input type="checkbox" checked={completed} onChange={onToggle} />

<span onClick={onToggle} style={{ marginLeft: '10px', cursor: 'pointer' }}>{text}</span>

<button onClick={onDelete}>Delete</button>

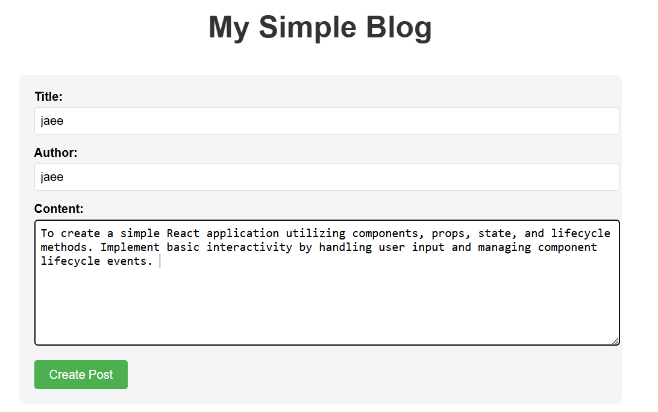
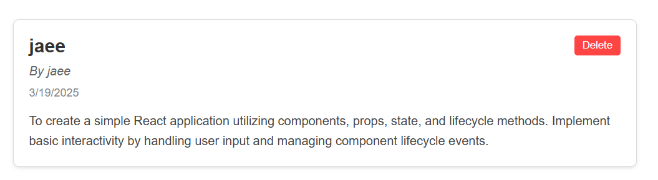
</li>

);

}

export default TodoItem;

OUTPUT:

Conclusion

This experiment provided an understanding of how to create a simple React application using components, props, state, and lifecycle methods. Additionally, handling user interactions through event handling was demonstrated, making the application interactive and dynamic.

| 10 |
| --- |