Aman Raj

Superset ID: 6358186

1. **ReactJS**

**Need and Benefits of Component Lifecycle**

* React components don’t just render once—they go through a lifecycle of creation, updating, and removal.
* The lifecycle methods let developers:

1. Run code at specific times, like before or after a component mounts or updates.
2. Optimize performance by controlling when to update or stop updates.
3. Manage side effects, e.g., fetching data when a component loads or cleaning up timers when it’s removed.

**Benefits:**

* Predictable component behavior.
* Cleaner code by separating logic for different phases.
* Better control of API calls, subscriptions, and DOM manipulation.

**Post.js**

class Post {

constructor(id, title, body) {

this.id = id;

this.title = title;

this.body = body;

}

}

export default Post;

**Posts.js**

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [],

error: null

};

}

// Method to fetch posts

loadPosts() {

fetch('https://jsonplaceholder.typicode.com/posts')

.then(response => response.json())

.then(data => {

const loadedPosts = data.map(p => new Post(p.id, p.title, p.body));

this.setState({ posts: loadedPosts });

})

.catch(error => {

this.setState({ error });

});

}

// Runs once after component is added to the DOM

componentDidMount() {

this.loadPosts();

}

// Handles any errors in rendering

componentDidCatch(error, info) {

alert('An error occurred: ' + error.message);

}

render() {

const { posts, error } = this.state;

if (error) {

return <h2>Error: {error.message}</h2>;

}

return (

<div>

<h1>Blog Posts</h1>

{posts.map(post => (

<div key={post.id}>

<h2>{post.title}</h2>

<p>{post.body}</p>

</div>

))}

</div>

);

}

}

export default Posts;

**App.js**

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div>

<Posts />

</div>

);

}

export default App;

**Output:-**

