Name: Tunuguntla Harshitha

Supersetid : 4992580

Email : 2200033132cseh@gmail.com

4.REACT-JS HOL:

1. Need and Benefits of Component Life Cycle
A component life cycle refers to the series of events (or phases) that a React component goes through from creation (mounting) to update (re-rendering) and finally removal (unmounting) from the DOM.
Need:

- To perform initialization tasks like data fetching or setting up subscriptions (when the component is created).
- To handle updates efficiently (e.g., re-render only when necessary).
- To clean up resources (like event listeners or timers) before the component is removed.
- To control and optimize the rendering process.
Benefits:
- Helps in better control over the component's behavior at different stages.
- Allows performance optimization (e.g., preventing unnecessary renders).
- Enables data fetching, DOM manipulation, and subscriptions at appropriate times.
- Facilitates resource cleanup and prevents memory leaks.
- Makes debugging easier by providing hooks to track the component’s lifecycle.

2. Various Life Cycle Hook Methods

In React (especially in Class Components), the life cycle methods are grouped into three main phases:

A. Mounting Phase (component is created and inserted into the DOM):

- constructor()
- static getDerivedStateFromProps()
- render()
- componentDidMount()

B. Updating Phase (component is re-rendered due to state/props changes):

- static getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

C. Unmounting Phase (component is removed from the DOM):

- componentWillUnmount()

3. Sequence of Steps in Rendering a Component

When a component is mounted:

1. constructor()
2. static getDerivedStateFromProps()
3. render()
4. componentDidMount()

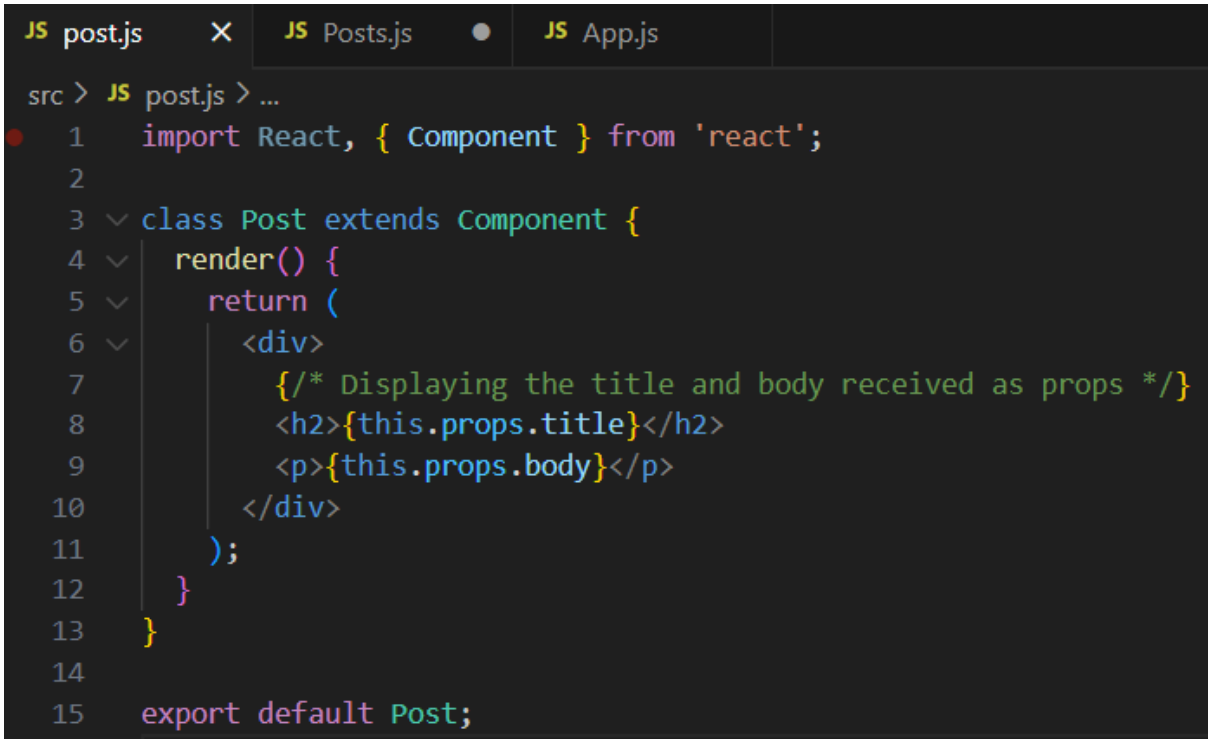
When a component updates (re-renders):

1. static getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

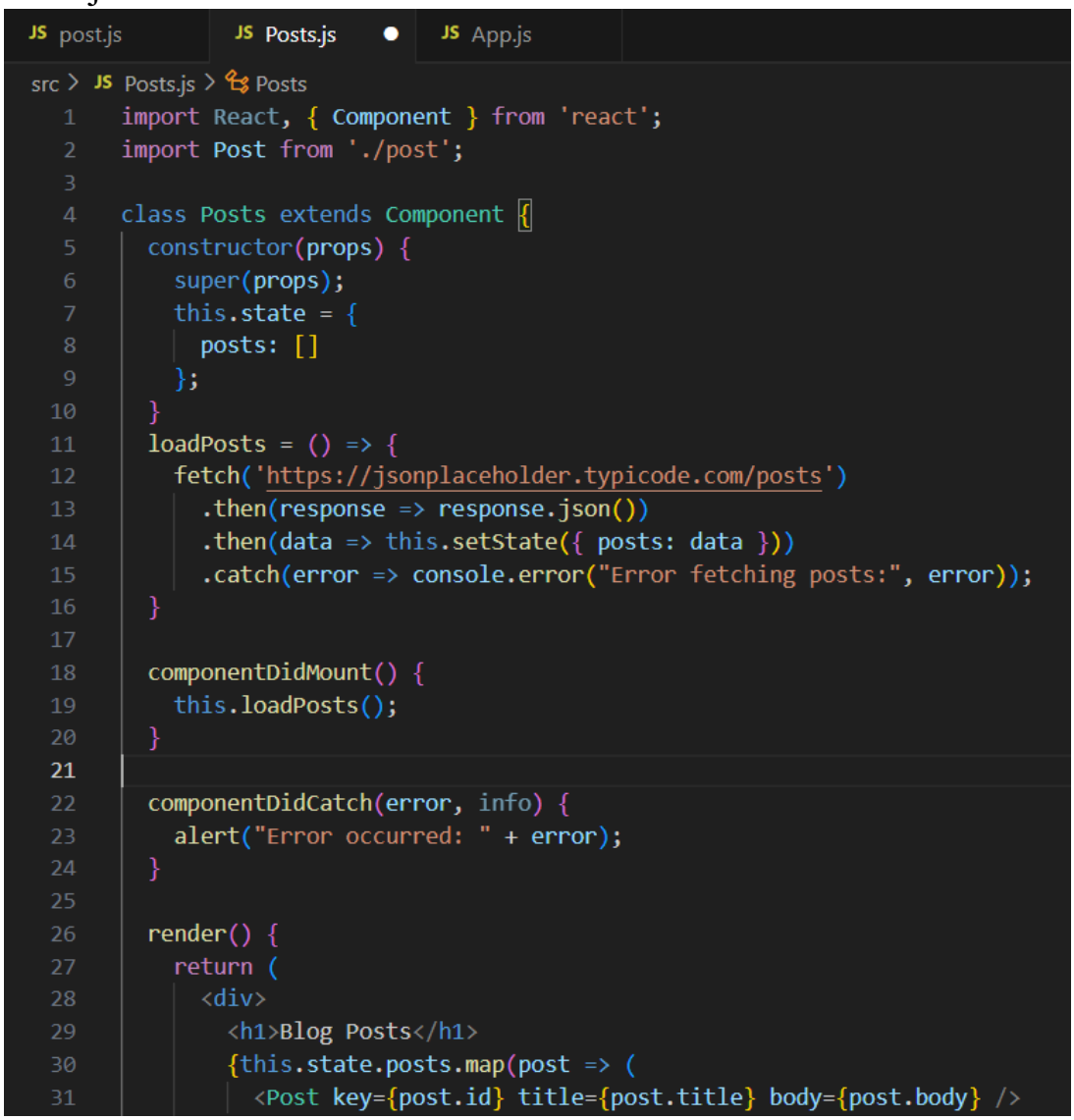
When a component unmounts:

1. componentWillUnmount()

Post.js:


post.js
Posts.js
App.js
src > post.js > ...
1 import React, { Component } from 'react';
2
3 class Post extends Component {
4 render() {
5 return (
6
7 {/* Displaying the title and body received as props */}
8 <h2>{this.props.title}</h2>
9 {this.props.body}
10
11);
12 }
13 }
14
15 export default Post;

Posts.js:


post.js
Posts.js
App.js
src > Posts.js > Posts
import React, { Component } from 'react';
import Post from './post';
class Posts extends Component {
constructor(props) {
super(props);
this.state = {
posts: []
};
}
loadPosts = () => {
fetch('https://jsonplaceholder.typicode.com/posts')
.then(response => response.json())
.then(data => this.setState({ posts: data }))
.catch(error => console.error("Error fetching posts:", error));
}
componentDidMount() {
this.loadPosts();
}
componentDidCatch(error, info) {
alert("Error occurred: " + error);
}
render() {
return (

<h1>Blog Posts</h1>
{this.state.posts.map(post => (
<Post key={post.id} title={post.title} body={post.body} />
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

App.js:

```
src > JS App.js > ...
1   import React from 'react';
2   import Posts from './Posts';
3
4   function App() {
5     return (
6       <div className="App">
7         <Posts />
8       </div>
9     );
10  }
11
12  export default App;
13
```

Output: