

Name: Ilam Venkata Satwik  
Super\_Set\_Id: 4992334  
Email: [2200032398cseh@gmail.com](mailto:2200032398cseh@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:

JS post.js



JS Posts.js



JS App.js

src > JS post.js > ...

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

## Posts.js:

```
JS post.js  JS Posts.js  JS App.js
src > JS Posts.js > Posts
1  import React, { Component } from 'react';
2  import Post from './post';
3
4  class Posts extends Component {
5    constructor(props) {
6      super(props);
7      this.state = {
8        posts: []
9      };
10   }
11   loadPosts = () => {
12     fetch('https://jsonplaceholder.typicode.com/posts')
13       .then(response => response.json())
14       .then(data => this.setState({ posts: data }))
15       .catch(error => console.error("Error fetching posts:", error));
16   }
17
18   componentDidMount() {
19     this.loadPosts();
20   }
21
22   componentDidCatch(error, info) {
23     alert("Error occurred: " + error);
24   }
25
26   render() {
27     return (
28       <div>
29         <h1>Blog Posts</h1>
30         {this.state.posts.map(post => (
31           <Post key={post.id} title={post.title} body={post.body} />
```

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:

