**Need and Benefits of Component Life Cycle**

Need

* In frameworks like React, components go through different phases: Mounting, Updating, and Unmounting.
* Each phase allows you to run specific code at the right moment, like:
  + Fetching data when the component appears.
  + Updating UI when props/state change.
  + Cleaning up resources before removing a component.
* Without life cycle methods, controlling these events manually would be complex and error-prone.

Benefits

1. Optimized Performance – You can perform tasks only when needed (e.g., API calls on mount).
2. Controlled Updates – Decide what happens when props or state change.
3. Resource Management – Clean up timers, event listeners, and memory leaks before unmounting.
4. Separation of Concerns – Each lifecycle phase has a clear responsibility.
5. Debugging and Maintenance – Easier to track what happens during rendering and updates.

**Various Life Cycle Hook Methods**

React lifecycle is divided into three phases:

A. Mounting (Component Creation)

* Methods triggered when the component is added to the DOM:
  + constructor()
  + static getDerivedStateFromProps()
  + render()
  + componentDidMount()

B. Updating (Re-rendering)

* Triggered when props or state changes:
  + static getDerivedStateFromProps()
  + shouldComponentUpdate()
  + render()
  + getSnapshotBeforeUpdate()
  + componentDidUpdate()

C. Unmounting (Component Removal)

* Triggered when component is removed from DOM:
  + componentWillUnmount()

D. Error Handling

* Triggered when there’s an error in rendering or lifecycle methods:
  + static getDerivedStateFromError()
  + componentDidCatch()

**Sequence of Steps in Rendering a Component**

When Component is Mounted (First Render)

1. constructor () – Initialize state and bind methods.
2. getDerivedStateFromProps() – Sync state with props before rendering.
3. render () – Return JSX to create UI.
4. componentDidMount() – Called after UI is inserted into DOM (API calls, subscriptions).

**Post.js**

// src/Post.js

import React from 'react';

class Post extends React.Component {

  render() {

    const { title, body } = this.props;

    return (

      <div style={{ border: "1px solid #ccc", margin: "10px", padding: "10px" }}>

        <h2>{title}</h2>

        <p>{body}</p>

      </div>

    );

  }

}

export default Post;

**Posts.js**

// src/Posts.js

import React from 'react';

import Post from './Post';

class Posts extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [

        {

          id: 1,

          title: "The Future of Web Development",

          body: "Modern frameworks like React and Next.js are shaping the way developers build fast and scalable applications."

        },

        {

          id: 2,

          title: "Understanding Component Lifecycle",

          body: "Mastering lifecycle methods is key to optimizing performance and handling side effects in React applications."

        },

        {

          id: 3,

          title: "Why Learn React Today",

          body: "React is one of the most popular JavaScript libraries, making it essential for frontend developers to know."

        }

      ],

      hasError: false

    };

  }

  componentDidCatch(error, info) {

    console.error("Error caught in component:", error, info);

    this.setState({ hasError: true });

    alert("An error occurred in the component!");

  }

  render() {

    if (this.state.hasError) {

      return <h2>Something went wrong while loading posts.</h2>;

    }

    return (

      <div>

        <h1>Blog Posts</h1>

        {this.state.posts.map(post => (

          <Post key={post.id} title={post.title} body={post.body} />

        ))}

      </div>

    );

  }

}

export default Posts;

**App.css**

.App {

  text-align: center;

  font-family: Arial, sans-serif;

}

h1 {

  color: #2c3e50;

}

div {

  margin: 20px auto;

  max-width: 600px;

}

**App.js**

// src/App.js

import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

  return (

    <div className="App">

      <Posts />

    </div>

  );

}

export default App;

**Output on terminal**

A screenshot of a computer program

AI-generated content may be incorrect.

**Output on the Browser**

A screenshot of a computer

AI-generated content may be incorrect.