**Explain the need and Benefits of component life cycle**

The Component Lifecycle in React refers to the series of stages a component goes through from the moment it is created to the moment it is removed (unmounted) from the DOM.

There are 3 main phases:

1. Mounting – Component is being created and inserted into the DOM.
2. Updating – Component is re-rendered due to changes in props or state.
3. Unmounting – Component is removed from the DOM.

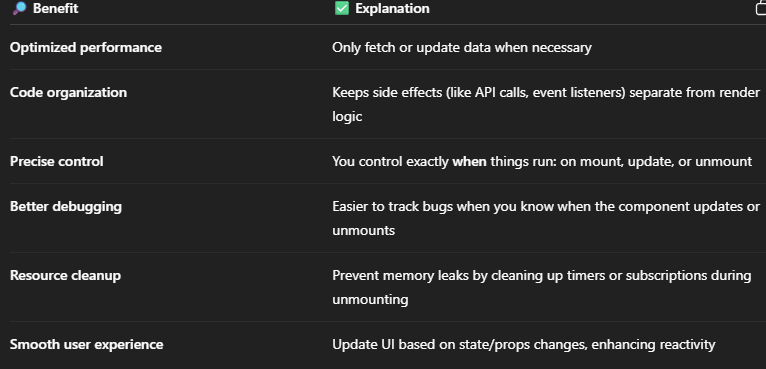
**Why Is the Component Lifecycle Needed?**

React components often need to:

* Fetch data from an API
* Start or stop timers
* Set up or clean up event listeners
* Save or load data (e.g., from localStorage)
* Control when and how things should update or reset

You cannot do all of this inside just the render() function. That’s why lifecycle methods/hooks like componentDidMount() or useEffect() exist.

**Benefits of Understanding the Component Lifecycle**



1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code

3.Create a new file named as **Post.js** in **src folder** with following properties

class Post {

  constructor(id, title, body) {

    this.id = id;

    this.title = title;

    this.body = body;

  }

}

export default Post;

4.Create a new class based component named as **Posts** inside **Posts.js** file

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

      error: null

    };

  }

  loadPosts = async () => {

    try {

      const response = await fetch('https://jsonplaceholder.typicode.com/posts');

      const data = await response.json();

      const postList = data.map(item => new Post(item.id, item.title, item.body));

      this.setState({ posts: postList });

    } catch (err) {

      this.setState({ error: err.message });

    }

  };

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

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

    console.error("Error info:", info);

  }

  render() {

    const { posts, error } = this.state;

    if (error) {

      return <h2 style={{ color: 'red' }}>Error: {error}</h2>;

    }

    return (

      <div>

        <h1>Blog Posts</h1>

        {posts.map(post => (

          <div key={post.id} style={{ marginBottom: '20px' }}>

            <h3>{post.title}</h3>

            <p>{post.body}</p>

            <hr />

          </div>

        ))}

      </div>

    );

  }

}

export default Posts;

5.Initialize the component with a list of Post in state of the component using the constructor

6.Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)

7.Add the Posts component to App component.

8.Build and Run the application using *npm start* command.

