1.Components Lifecycle and State Management

In React.js, lifecycle methods are functions that get executed at various stages in the lifecycle of a component. There are three main phases: Mounting, Updating, and Unmounting. Here are some key lifecycle methods:

1. **Mounting Phase:**

   - `constructor`: Initializes the component.

   - `render`: Outputs the UI representation.

   - `componentDidMount`: Invoked after the component is rendered to the DOM.

2. **Updating Phase:**

   - `shouldComponentUpdate`: Determines if the component should re-render.

   - `render`: Re-renders the component.

   - `componentDidUpdate`: Called after the component has been updated.

3. **Unmounting Phase:**

   - `componentWillUnmount`: Called just before the component is removed from the DOM.

Here’s a simple React application demonstrating the use of lifecycle methods and state management:

```jsx
Import React, { Component } from ‘react’;

Class LifecycleDemo extends Component {
 Constructor(props) {
  Super(props);
  This.state = { message: ‘Hello, React!’ };
 }
```

```
  componentDidMount() {
    // Side effect: Triggered after the component is rendered
    Console.log('Component is mounted');
  }

  componentDidUpdate(prevProps, prevState) {
    // Side effect: Triggered after the component updates
    Console.log('Component is updated');
  }

  componentWillUnmount() {
    // Cleanup: Triggered just before the component is removed
    Console.log('Component will unmount');
  }

  handleClick = () => {
    // Update state when a button is clicked
    This.setState({ message: 'Updated message!' });
  };

  Render() {
    Return (
      <div>
        <p>{this.state.message}</p>
        <button onClick={this.handleClick}>Update Message</button>
      </div>
    );
  }
}
```

```
Export default LifecycleDemo;
```

This example includes a component with a message in its state. Clicking the “Update Message” button triggers a state update, which in turn invokes the `componentDidUpdate` lifecycle method. The console logs in each lifecycle method illustrate their respective triggers.

Remember to use class components for lifecycle methods if you’re not using React Hooks. If you’re working with functional components, you can use the `useEffect` hook for similar effects.