MODULE: 10 List and Hooks

* **Explain Life cycle in Class Component and functional component with Hooks.**

Ans) there are two types of components:

1)Class Components

2) Functional Components.

**Class Components:**

Class components are the traditional way of creating components in React.

They extend the React.Component class and have access to lifecycle methods that are called at different stages of the component's lifecycle.

**Lifecycle Methods in Class Components:**

**Mounting**

constructor (): Called before the component is mounted.

static getDerivedStateFromProps (): Called right before rendering the component.

render (): Required method that returns the elements to be rendered.

componentDidMount (): Called immediately after the component is mounted (inserted into the tree).

**Updating**

static getDerivedStateFromProps (): Called right before re-rendering when new props are received.

shouldComponentUpdate (): Called before re-rendering when new props or state are received.

render (): Required method that returns the elements to be rendered.

getSnapshotBeforeUpdate (): Called right before the changes from render () are committed to the DOM.

componentDidUpdate (): Called immediately after the component is updated.

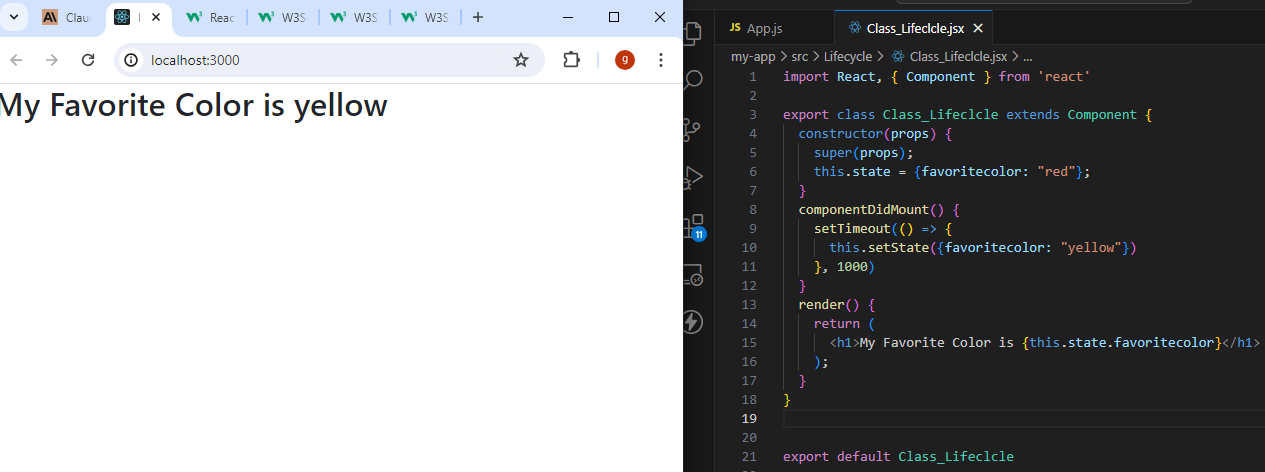
**Unmounting**

componentWillUnmount (): Called immediately before the component is unmounted and destroyed.

**Error Handling**

static getDerivedStateFromError (): Called after an error has been thrown by a descendant component.

componentDidCatch (): Called after an error has been thrown by a descendant component.



**Functional Components with Hooks:**

Functional components are the modern way of creating components in React. They are simpler and more lightweight than class components. However, they lacked state management and lifecycle methods until the introduction of React Hooks.

Hooks in Functional Components:

React Hooks provide a way to manage state and lifecycle methods in functional components. Some of the commonly used hooks are:

**State Hooks**

useState (): Used for managing state in functional components.

**Effect Hooks**

useEffect (): Used for performing side effects in functional components. It combines the lifecycle methods componentDidMount, componentDidUpdate, and componentWillUnmount.

**Context Hooks**

useContext (): Used for accessing context values in functional components.

**Other Hooks**

use Reducer (): Used for managing state with a reducer function.

useCallback (): Used for memorizing a callback function.

use Memo (): Used for memorizing a value.

useRef (): Used for creating a mutable reference to a value or DOM node.

