React- Day -4: React components:

Component life cycle:

React 16.8 has led to changes in the traditional lifecycle methods used in class components.

Class Component Lifecycle Methods:

Mounting Phase: called **before every render** when new **props or state are received**.

Updating Phase : Called **after the component's updates**

Unmounting Phase : Called immediately before a **component is unmounted or destroyed**.

Functional Component Lifecycle with Hooks:

Mounting Phase: useState(), useEffect()

Updating Phase : useState(), useEffect()

(useMemo() and useCallback() **AVOIDING UNNECESSARY RE RENDERING THE COMPONENTS**)

Unmounting Phase : useEffect():

Introduction to Hook:

React state and lifecycle features from **functional components**, instead of using class components. They were introduced in React 16.8 as a way to simplify state management and side effects in functional components.

Major or familiar hooks:

useState, useeffect, useContext

Other Hooks:

useCallback, useMemo, useRef, useReducer

UseState :

hook is one of the fundamental React hooks that allows you to add state to functional components.

The useState hook takes an initial value as its argument and returns an array containing two elements: the **current state value** and a **function to update that value.**

const [stateValue, setStateFunction] = useState(initialValue);

UseEffect & dependency Array :

that allows you to **perform side effects in functional components**. Side effects include actions like data fetching, DOM manipulation.

Dependency array : Mounting , Updating, Unmounting

[] – empty array -> render once

[props,state] - > update

Return -> clear

Note: Flipkart or amazon opening screen

Stateful and stateless components:

Stateful and stateless components are terms used in React to describe **components based on whether they manage state or not.**

Functional Components: Also known as stateless components till React 16.8 version.

in React 16.8, **allow functional components to manage state using the useState and other hook functions.**

Eg: stateless component:

```
import React from 'react';
const StatelessComponent = ({a}) => {
    return <div>{a}</div>;
    };
export default StatelessComponent;
```

Class Components (Traditional): Also known as stateful components.

Eg: state component:

```
import React, { Component } from 'react';
class StatefulComponent extends Component {
    constructor(props) {
    super(props);
    this.state = {
        message: 'Hello, Stateful Component!',
        };
     }

 render() {
    return <div>{this.state.message}</div>;
     }
    }
export default StatefulComponent;
```

Practical Example

Reusable components :

Reusable components are a fundamental concept in React, allowing developers to create **modular and maintainable code**. Reusable components can be used across different parts of an application, promoting **code reusability and reducing redundancy.**

Passing dynamic data to component:

It can be achieved by using **props**. Props (short for properties) allow you to pass data from a parent component to a child component.

Using **state**, allows you to manage and update dynamic data within a component.

Components - In-depth: **Practical Example**