Lifecycle of Components In Class- Components

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.

The three phases are: **Mounting**, **Updating**, and **Unmounting**.

Mounting :

The constructor method is the very first method called during the mounting phase.

Mounting means putting elements into the DOM.

Mounting is the phase of the component lifecycle when the initialization of the component is completed and the component is mounted on the DOM and rendered for the first time on the webpage. Now React follows a default procedure in the Naming Conventions of these predefined functions where the functions containing “Will” represents before some specific phase and “Did” represents after the completion of that phase. The mounting phase consists of two such predefined functions as described below.

- **componentWillMount() Function**: As the name clearly suggests, this function is invoked right before the component is mounted on

the DOM i.e. this function gets invoked once before the render() function is executed for the first time.

- **componentDidMount() Function**: Similarly as the previous one this function is invoked right after the component is mounted on the DOM i.e. this function gets invoked once after the render() function is executed for the first time

Updating :

The next phase in the lifecycle is when a component is *updated*.

A component is updated whenever there is a change in the component's state or props.

The componentDidUpdate method is called after the component is updated in the DOM.

componentDidUpdate() Function: this function is invoked after the component is rerendered i.e. this function gets invoked once after the render() function is executed after the updation of State or Props.

Unmounting :

This is the final phase of the lifecycle of the component that is the phase of unmounting the component from the DOM.

- **componentWillUnmount() Function**: This function is invoked before the component is finally unmounted from the DOM i.e. this function gets invoked once before the component is removed from the page and this denotes the end of the lifecycle.

Lifecycle of Components In Function- Components

we can use lifecycle methods in functional components with the help of the useEffect() hook. we can achieve functionality similar to lifecycle methods in the functional components using react hooks.

The useEffect() in react is used to manage the side effects. It is one of the most commonly used react hooks. The syntax of the useEffect hook is as follows:

```
useEffect( () => {

}, [ ])
```

- By using this hook, we can create functionality similar to componentDidMount, componentDidUpdate, and componentWillUnmount.

- useEffect works same as componentDidMount It holds the side effects, and only runs once, when the component is mounted. If you add dependencies in the dependency array, the function passed into the useEffect hook will run every time the dependencies, like a piece of stateful data, changes. This behavior of the useEffect hook is comparable to the componentDidUpdate method since it's invoked on every state/props change.

- We have to return a function from the useEffect() hook. This function will execute only when the component is removed from the DOM tree, thus making it similar to the componentWillUnmount lifecycle method.