**Mounting & Unmounting Components**

In React, when we talk about a component "mounting" or "unmounting," we're referring to specific stages in the component's lifecycle. Here's a detailed explanation:

1. **Mounting**:
   * This is the phase in which the component is being created and inserted into the DOM.
   * When a component is mounted, it means the component's output is rendered for the first time and its DOM representation is attached to the page.
   * Several lifecycle methods are called during this phase (in the context of class components). The typical order of their execution is: **constructor** -> **static getDerivedStateFromProps** (if used) -> **render** -> **componentDidMount**.
   * **componentDidMount** is the most commonly used mounting lifecycle method. It's called after the component's output has been rendered to the DOM. It's a good place to initiate network requests or set up subscriptions, timers, and other side-effects.
   * In functional components, the **useEffect** hook with an empty dependency array **[]** can mimic the behavior of **componentDidMount**:

useEffect(() => {

// Code here will run once, similar to componentDidMount

}, []);

1. **Unmounting**:
   * This is the phase in which the component is being removed from the DOM.
   * When a component is unmounted, it means the component is about to be destroyed and removed from the rendered output.
   * In class components, **componentWillUnmount** is the lifecycle method that's called just before the component is removed from the DOM. It's a good place to clean up any side-effects, clear timers, cancel network requests, and revoke any subscriptions to avoid memory leaks.
   * In functional components, the **useEffect** hook allows you to return a cleanup function, which serves a similar purpose as **componentWillUnmount**:

useEffect(() => {

// Code here will run once, similar to componentDidMount

return () => {

// Cleanup code here will run when the component is about to unmount

};

}, []);

It's crucial to understand these lifecycle stages, especially when working with side effects, third-party libraries, or any code that needs to interact with the broader environment outside the component. Properly managing resources during mounting and unmounting helps prevent potential bugs, memory leaks, and ensures smooth performance.