React Functional Component Lifecycle with useEffect

Overview

In functional components, we use the useEffect Hook to replicate lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount found in class components.

Basic Syntax of useEffect

```
useEffect(() => {
  // code to run on mount/update
  return () => {
    // cleanup code
  };
}, [dependencies]);
```

Lifecycle Mappings

Class Component -> Functional Hook Equivalent -> When it Runs

componentDidMount() -> useEffect(() => {}, []) -> After first render

componentDidUpdate() -> useEffect(() => {...}, [dep]) -> When dependency changes

componentWillUnmount() -> useEffect(() => { return () => {...}; }, []) -> On unmount

Example 1: componentDidMount

```
useEffect(() => {
  console.log('Mounted');
  // e.g., fetch API
}, []);
```

Example 2: componentDidUpdate

```
useEffect(() => {
  console.log('Updated count:', count);
}, [count]);
```

React Functional Component Lifecycle with useEffect

Example 3: componentWillUnmount

```
useEffect(() => {
  const interval = setInterval(() => console.log('Running...'), 1000);
  return () => {
    clearInterval(interval);
    console.log('Component Unmounted');
  };
}, []);
```

Full Lifecycle Example

```
useEffect(() => {
  console.log('Mounted');
  return () => console.log('Unmounted');
}, []);
useEffect(() => {
  if (count > 0) console.log('Updated count:', count);
}, [count]);
```

Understanding Unmount

Unmounting means the component is removed from the DOM (e.g., switching pages).

The cleanup function in useEffect is executed before unmount or before re-running effect.

Analogy: You set up a fan in a room (component). When leaving the room (unmount), you turn off the fan (cleanup).

Code:

```
function Timer() {
  useEffect(() => {
    const id = setInterval(() => console.log('Running...'), 1000);
    return () => {
```

```
      console.log('Cleaning up');
      clearInterval(id);
    };
  }, []);
  return <h2>Timer</h2>;
}
```

In <App>, toggling Timer with a button causes unmount and triggers cleanup.

Summary Table

Lifecycle Stage -> Functional useEffect

On Mount -> useEffect(() => {}, [])

On Update -> useEffect(() => {...}, [dep])

On Unmount -> useEffect(() => { return () => {...} }, [])