React Side Effects – Beginner Guide

What is a Side Effect?

In React (and programming), a side effect is any action that happens outside the scope of the current function’s direct return value — it affects something external or depends on something external.
If your function does more than just calculate and return a value — it’s doing a side effect.

Examples of Side Effects in React

Example	**Why It's a Side Effect?**
Fetching data from an API	Involves network request → outside the component.
Setting up subscriptions or event listeners	Component affects browser’s global state (like window events).
Changing the DOM directly	React manages DOM itself, so manual DOM changes are 'outside' work.
Setting a timer (setTimeout, setInterval)	Interacts with browser timer API.
Logging analytics or tracking	Sends info to external systems.
WebSocket connections	Communicating outside the component.

What is NOT a Side Effect?

- Updating component state during render based only on props/state.
- Calculating derived values from props/state (useMemo).
- Rendering JSX itself.

Why useEffect is for Side Effects?

React’s render process should be pure — same input → same output. Side effects break purity because:
- They may depend on external state (network, DOM, browser).
- They may cause external changes (API calls, DOM mutations).
So React says: render first, then run side effects in useEffect so UI updates are consistent.

Real-time Example

Imagine you open a Blinkit-like cart page:
1. Render: Show cart UI from state.
2. Side effect: Fetch updated stock status from server → update state → re-render.