React useEffect — Good vs Bad Usage + Diagram + Do & Don’t Guide

1. Visual Diagram: Render vs Effect

RENDER PHASE (Pure) ■ ■ • Should NOT cause side effects ■ • Should ONLY return UI ▼ UI IS DISPLAYED ■ ▼ EFFECT PHASE (useEffect) ■ ■ • Runs AFTER UI appears ■ • Safe for side effects: ■ - API calls ■ - document.title ■ - timers ■ - localStorage ■ - subscriptions ▼ CLEANUP (before next effect or on unmount)

2. Good vs Bad useEffect Usage

Bad Practice	Why It's Wrong	Correct Practice
document.title inside render	Side effect during render	Use useEffect(() => { document.title = ... }, [count])
API fetch inside render	Runs every render → infinite loop	Use useEffect(() => fetch(), [])
setInterval in render	Creates multiple timers	Use useEffect with cleanup
Adding event listener in render	Duplicates listeners every render	Use useEffect with cleanup
localStorage inside render	Runs too often	Use useEffect([dependencies])

3. Good Example 1: Update Title Correctly

```
useEffect(() => { document.title = `Clicked ${count} times`; }, [count]);
```

4. Bad Example 1: Updating Title in Render (Incorrect)

```
document.title = `Clicked ${count} times`; // ■ not allowed inside render
```

5. Good Example 2: Fetch API on Mount

```
useEffect(() => { fetch('/api/users') .then(r => r.json()) .then(setUsers); }, []);
```

6. Bad Example 2: Fetch API Inside Render

```
const data = fetch('/api/users'); // ■ infinite calls — very wrong
```

7. Good Example 3: Timer with Cleanup

```
useEffect(() => { const timer = setInterval(() => console.log("Tick"), 1000); return () =>
clearInterval(timer); }, []);
```

8. Bad Example 3: Timer Inside Render

setInterval(() => console.log("Tick"), 1000); // ■ creates unlimited timers

Do & Don’t List (Very Important)

■ Do (Correct Use)

• Use useEffect for side effects • Use it for API calls • Use it for document.title updates • Use it for timers • Use it for subscriptions/event listeners • Use cleanup to avoid memory leaks

■ Don't (Incorrect Use)

• Don’t put side effects inside render • Don’t put API fetch directly in component body • Don’t create timers in render • Don’t modify DOM inside render • Don’t write to localStorage inside render

Summary

Rendering must remain PURE. Side effects must go inside useEffect. Cleanup prevents memory leaks.