# useEffect in React – Master Guide

## What is useEffect?

The useEffect hook lets you **perform side effects** (like API calls, event listeners, timers, etc.) in function components.

```
useEffect(() => {
  // your code here
}, [dependencies]);
```

## Component Lifecycle Breakdown in React

| Lifecycle Phase | Description |
| --- | --- |
| ☑ Mount | Component loads for the **first time** |
| 🔁 Re-render | Component **updates** due to state/props change |
| ❌ Unmount | Component is **removed** from UI |

## useEffect Variants & Their Behavior

### ☑ 1. **Empty Dependency Array** []

```
useEffect(() => {
  console.log("🎯 Runs only once (on mount)");
}, []);
```

- 🔂 Called only **once**, when the component **mounts**.
- ❌ Not called on re-renders.
- ☑ Good for one-time things (e.g., fetching data on load).

> 🧠 **Similar to** componentDidMount() in class components.

### 🔁 2. **With Specific Dependencies** [value]

```
useEffect(() => {
  console.log("🔁 Runs when value changes");
}, [value]);
```

- 🩶 Called on **mount** ☑.
- 🔁 Runs again **only when** `value` **changes**.
- ❌ Not on every re-render (unless `value` is changing).

> 🧠 **Similar to** `componentDidUpdate()` for specific props/state.

---

### ♻️ 3. **No Dependency Array**

```
useEffect(() => {
  console.log("♻️ Runs after every render");
});
```

- 🔁 Runs after **every render** — both mount and all re-renders.
- ⚠️ Use this **rarely** to avoid performance issues.

---

### 💥 4. **Cleanup Function (Unmount Handling)**

```
useEffect(() => {
  console.log("👋 Mounted");

  return () => {
    console.log("🧹 Cleaned up before unmount or next run");
  };
}, []);
```

- ☑ Runs on mount.
- 🧹 Cleanup function runs when:
  - Component **unmounts**.
  - OR before the **next effect** if dependencies change.

---

## 🎯 Full Example With All Cases

```
import React, { useState, useEffect } from "react";

function DemoComponent() {
  const [count, setCount] = useState(0);

  // 🧠 Mount only
  useEffect(() => {
    console.log("🚀 Mounted once");

    return () => {
```

```
      console.log("🧹 Component is unmounting");
    };
  }, []);

  // 🔁 Only when count changes
  useEffect(() => {
    console.log("📈 count changed:", count);
  }, [count]);

  // ♻️ Every render
  useEffect(() => {
    console.log("♻️ I run after every render");
  });

  return (
    <div>
      <h2>Count: {count}</h2>
      <button onClick={() => setCount(count + 1)}>➕ Increment</button>
    </div>
  );
}
```

## 🧠 Difference Between Re-render and Unmount

| Feature | Re-render | Unmount |
|---|---|---|
| When? | State/props change | Component removed from UI |
| `useEffect([])` | ❌ Not triggered again | ❌ Not triggered |
| `useEffect([x])` | ☑️ If `x` changes | ❌ Not triggered unless cleaned up |
| Cleanup runs? | ☑️ If deps changed (before re-run) | ☑️ Yes, on unmount |

## 🔧 Practical Use Cases

| Use Case | Hook Example |
|---|---|
| Fetch API on mount | `useEffect(() => { fetch() }, [])` |
| Update on input change | `useEffect(() => { console.log(x) }, [x])` |
| Clean interval/timer | `useEffect(() => { const t = setInterval...; return () => clearInterval(t) }, [])` |
| Socket disconnect | `return () => socket.disconnect();` |

## ⚠️ Common Mistakes

❌ Forgetting to add dependencies → effect doesn't update correctly. ☑ Always include **everything used inside `useEffect`** in the dependency array.

---

## Bonus: Dependency Array Visualization

```
useEffect(() => {
  console.log("Runs ONLY when a OR b changes");
}, [a, b]);
```

- ☑ Runs when:
  - `a` changes
  - `b` changes
  - OR initial mount
- ❌ Does NOT run on:
  - Unrelated state changes

---

## Summary Cheat Sheet

| Syntax | Triggers On | Cleanup Runs On |
|---|---|---|
| `useEffect(() => {}, [])` | ☑ Mount | ❌ Unmount only |
| `useEffect(() => {}, [x])` | ☑ Mount, 🔁 `x` change | ☑ Before re-run / unmount |
| `useEffect(() => {})` | 🔁 Every render | ☑ Before every next run |
| `return () => {}` | ❌ Not called immediately | ☑ On unmount OR next effect |

---