Start with: **What is useEffect?**

✓ useEffect is a React Hook used to run **side effects** ✓ Side effects = Anything that happens **outside the UI** Examples:

- Fetch API data
- Working with localStorage
- Timers (setTimeout, setInterval)
- Event listeners
- Updating document title

> 🔑 It runs **after render**.

📌 Basic Structure:

```
useEffect(() => {
  // Side effect code here
});
```

2️⃣ Explain: When Does useEffect Run?

There are **3 main modes**:

useEffect Syntax	When it Runs	Usage
`useEffect(fn)`	Every render	Debugging or event tracking (rare)
`useEffect(fn, [])`	Only once (first render)	Fetch API, initial setup
`useEffect(fn, [state])`	When given state/prop changes	Depends on specific event

🧪 Example 1 — useEffect without dependency

(Side effect runs every time component renders)

Example: console log on every render

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

export default function App() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    console.log("useEffect Running: Every Render");
  });

  return (
    <div>
```

```
      <h1>Count: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

🧠 Teaching Point:

- After every button click → rerender → useEffect runs again

🧪 Example 2 — useEffect with empty dependency `[]`

(Only one time like **componentDidMount**)

Example: Update document title once

```
useEffect(() => {
  document.title = "Welcome to Bootcamp!";
}, []);
```

🧠 Teaching Point: ✓ Perfect for initial API calls ✓ Setup things once

---

🧪 Example 3 — useEffect with Dependency (Event-based)

(run only when value changes)

Example: Update title only when count changes

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

export default function App() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `Clicked ${count} times`;
  }, [count]); // 👈 Runs only when count updates

  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Click</button>
    </div>
  );
}
```

🧠 Teaching Point:

- The "event" is **count state change**
- React sees `count` updated → useEffect runs

🎯 Explain Cleanup Function

Some effects require cleanup (unsubscribe, clear timers)

Example: removing setInterval

```
useEffect(() => {
  let timer = setInterval(() => {
    console.log("Running...");
  }, 1000);

  return () => {
    clearInterval(timer);
    console.log("Timer stopped");
  };
}, []);
```

🧠 Teaching Point: ✓ Cleanup runs when component is removed ✓ Prevents memory leaks

---

🧠 Final Summary Table

Type	**Code**	**When It Runs**	**Best Use Case**
No Dependency	`useEffect(fn)`	Every render	Debugging, Logging
Empty Dependency	`useEffect(fn, [])`	Only once	Fetch data initially
With Dependency	`useEffect(fn, [value])`	When value changes	Event-based effects

💡 Small Quiz for Students

Ask them:

> If useEffect has `[name, age]` as dependency, when will it run?

Expected answer:

> Whenever name OR age changes
