useEffect

1) WHAT is useEffect?

- useEffect is a **React Hook**.
- It lets you run **side effects** in functional components.
- Side effects = work that happens **outside** the normal UI rendering.

Examples of side effects:

- Fetch API calls
- setInterval, setTimeout
- Adding event listeners
- Updating document.title
- Storing data in localStorage
- Subscriptions (sockets, auth, etc.)

---

2) WHY we use useEffect?

Because React components are **pure** functions.
Rendering → should NOT:

- fetch data
- access browser API
- run timers
- change DOM manually
- make network calls

So React gives a separate place — **useEffect** — to safely do side-effects.

Why?

1. Keeps UI rendering pure.
2. Avoids infinite loops.
3. Controls WHEN side effects run.
4. Allows cleanup (important for timers/listeners).
5. Improves performance.

---

3) HOW does useEffect work?

Basic structure:

```
useEffect(() => {
  // side effect code here


  return () => {
    // cleanup (optional)
  };
}, [dependencies]);
```

Meaning:

- React **runs** the effect after render.
- If dependencies change → React runs the effect again.
- Before running again, React runs the **cleanup**.

Cleanup used for:

- Removing event listeners
- Clearing intervals

- Canceling network calls

---

4) WHEN does useEffect run?

This is the most important part.

Case 1 — No dependency array

useEffect(() => {});

Runs **after EVERY render**
(including re-renders)

Case 2 — Empty array []

useEffect(() => {}, []);

Runs **only once**
(when the component mounts)

Case 3 — With dependencies

useEffect(() => {}, [count]);

Runs **only when count changes**

Case 4 — Multiple dependencies

useEffect(() => {}, [a, b, c]);

Runs when **any** of a, b, c changes.

---

5) WHAT is "dependency array"?

The part inside:

[dependencies]

React checks:
Did any dependency change compared to previous render?
→ If **yes**, run effect again.

Purpose of dependency array:
✓ control effect timing
✓ avoid infinite loops
✓ avoid unnecessary reruns
✓ keep effect synchronized with state/props

---

6) WHY dependency array is important?

Because without it, your code will run:

- too many times
- or not at all
- or cause infinite loops
- or fetch API 50 times
- or create memory leaks

Correct dependency array = correct behavior.

---

7) HOW many types of effects exist?

1) Run on mount (one time)

```
useEffect(() => {
  console.log("Component mounted");
}, []);
```

2) Run on state change

```
useEffect(() => {
```

```
  console.log("Count changed");
}, [count]);
```

3) Run on every render (rare)

```
useEffect(() => {
  console.log("Every render");
});
```

4) Cleanup effect

```
useEffect(() => {
  const id = setInterval(...);

  return () => clearInterval(id);
}, []);
```

---

9) WHAT is a stale closure?

Effect stores **old values** if dependencies are incorrect.

Wrong:

```
useEffect(() => {
  console.log(count); // old count
}, []);
```

Add dependency:

```
useEffect(() => {
  console.log(count);
```

}, [count]);

---

12) WHEN NOT to use useEffect?

Avoid useEffect in:

Deriving state (can compute directly)

Reading props to setState (anti-pattern)

Running synchronous logic (use directly, not effect)

Transforming data (useMemo better)

---

13) What is inside useEffect allowed?

You can:

✓ Fetch data

✓ console.log

✓ timers

✓ listeners

✓ localStorage

✓ DOM manipulation

✓ update state (careful)

You CANNOT:

❌ Put async directly (use async function inside)

❌ Return non-function from effect

❌ Use it conditionally

---

14) useEffect CLEAN RULES

1. Effects should run after render

2. Cleanup should undo the effect
3. Dependency array MUST include all external values
4. Should be placed at top level

useEffect syntax -24 times

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

const useEffect = () => {

   useEffect(() => {
      console.log("useEffect 1 running");

   }, [dependancies])

   useEffect(() => {
      console.log("useEffect 2 running");
   }, [dependencies]);

   useEffect(() => {
      console.log("useEffect 3 running");

   }, [dependancies])

   useEffect(() => {
```

```
    console.log("useEffect 4 running");
}, [dependencies]);

useEffect(() => {
    console.log("useEffect 5 running");

}, [dependancies])

useEffect(() => {
    console.log("useEffect 6 running");
}, [dependencies]);

useEffect(() => {
    console.log("useEffect 7 running");

}, [dependancies])

useEffect(() => {
    console.log("useEffect 8 running");
}, [dependencies]);

useEffect(() => {
    console.log("useEffect 9 running");
```

```
}, [dependancies])

useEffect(() => {
   console.log("useEffect 10 running");
}, [dependencies]);

useEffect(() => {
   console.log("useEffect 11 running");

}, [dependancies])

useEffect(() => {
   console.log("useEffect 12 running");
}, [dependencies]);

useEffect(() => {
   console.log("useEffect 13 running");

}, [dependancies])

useEffect(() => {
   console.log("useEffect 14 running");
```

```
}, [dependencies]);

useEffect(() => {
   console.log("useEffect 15 running");

}, [dependancies])

useEffect(() => {
   console.log("useEffect 16 running");
}, [dependencies]);

useEffect(() => {
   console.log("useEffect 17 running");

}, [dependancies])

useEffect(() => {
   console.log("useEffect 18 running");
}, [dependencies]);

useEffect(() => {
   console.log("useEffect 19 running");
```

```
    }, [dependancies])

    useEffect(() => {
        console.log("useEffect 20 running");
    }, [dependencies]);

    useEffect(() => {
        console.log("useEffect 21 running");
    }, [dependancies])

    useEffect(() => {
        console.log("useEffect 22 running");
    }, [dependencies]);

    useEffect(() => {
        console.log("useEffect 23 running");
    }, [dependancies])

    useEffect(() => {
        console.log("useEffect 24 running");
    }, [dependencies]);

}
```