React useRef Hook

---

1. Simple Meaning of useRef

`useRef` is a React hook used to **store a value that should not refresh the screen when it changes**.

Very important idea:

- useState → updates UI (screen changes)
- useRef → does NOT update UI (only logic changes)

So React remembers the value, but does not redraw the component.

---

2. Syntax

```
const myRef = useRef(initialValue);
```

It returns an object like:

```
{
  current: initialValue
}
```

We always read/write using:

```
myRef.current
```

---

3. Why we need useRef?

Sometimes we must remember data but it is NOT part of UI. If we store it in state → React re-renders again and again (slow performance).

So React gives useRef.

We use it for:

- timers
- interval id
- previous value
- input focus

- scroll position
- storing DOM elements
- tracking API calls

---

4. Example — Access DOM Element (Focus Input)

```
import React, { useRef } from 'react';

function FocusInput() {
  const inputRef = useRef(null);

  const focusInput = () => {
    inputRef.current.focus();
  };

  return (
    <div>
      <input ref={inputRef} type="text" />
      <button onClick={focusInput}>Focus Input</button>
    </div>
  );
}

export default FocusInput;
```

Explanation:

- React gives direct access to input element
- Works like document.querySelector
- No re-render happens

---

5. Example — Timer (Very Important)

Problem: setInterval returns an ID. We must store it to stop timer.

If we use state → component re-renders every second ❌ So we use useRef ✅

```
import React, { useRef, useState, useEffect } from 'react';

function Timer() {
  const [seconds, setSeconds] = useState(0);
  const intervalRef = useRef(null);

  useEffect(() => {
    intervalRef.current = setInterval(() => {
      setSeconds(prev => prev + 1);
    }, 1000);

    return () => clearInterval(intervalRef.current);
  }, []);

  return (
    <div>
      <p>Seconds: {seconds}</p>
      <button onClick={() => clearInterval(intervalRef.current)}>
        Stop Timer
      </button>
    </div>
  );
}

export default Timer;
```

Why useRef works here:

- stores interval id
- persists between renders
- no extra rendering
- allows stopping timer

---

6. useState vs useRef

Feature	useState	useRef
Causes re-render	Yes	No
Used for UI	Yes	No
Stores mutable value	No	Yes
Good for timers	No	Yes
Access DOM	No	Yes

---

7. When should you use useRef?

Use useRef when value is needed for logic but NOT for display.

Examples:

- timer id
- previous counter value
- checking first render
- focus input field
- scroll tracking

- storing websocket instance

---

8. Memory Trick

useState → show data on screen
useRef → remember data internally

9. Real Life Analogy

React Component = Classroom

useState = writing on board (everyone sees change) useRef = notes in teacher pocket (only teacher uses)

---

10. Interview Important Point

React re-renders when state changes. But timers, DOM access and background values should NOT re-render.

So in machine coding problems (Stopwatch, Carousel, Games), **useRef is used to store interval ID.**

---

11. Stopwatch Example (Most Asked Interview Question)

Features:

- Start timer
- Stop timer
- Reset timer
- Show time in HH\:MM\:SS

Code

```
import React, { useState, useRef } from 'react';

const Stopwatch = () => {
  const [time, setTime] = useState(0);
  const [isRunning, setIsRunning] = useState(false);
  const timerRef = useRef(null);

  const startTimer = () => {
    if (!isRunning) {
      setIsRunning(true);
      timerRef.current = setInterval(() => {
        setTime(prev => prev + 1);
      }, 1000);
    }
  };

  const stopTimer = () => {
    clearInterval(timerRef.current);
    setIsRunning(false);
  };

  const resetTimer = () => {
    clearInterval(timerRef.current);
    setIsRunning(false);
    setTime(0);
  };

  const formatTime = (time) => {
    const seconds = `0${time % 60}`.slice(-2);
    const minutes = `0${Math.floor(time / 60) % 60}`.slice(-2);
    const hours = `0${Math.floor(time / 3600)}`.slice(-2);
    return `${hours}:${minutes}:${seconds}`;
  };

  return (
    <div>
      <h1>{formatTime(time)}</h1>
      <button onClick={startTimer}>Start</button>
      <button onClick={stopTimer}>Stop</button>
      <button onClick={resetTimer}>Reset</button>
    </div>
  );
};

export default Stopwatch;
```

Why useRef is important here?

We store `setInterval` ID inside `timerRef.current`

Because:

- we must stop timer later
- component re-renders every second
- but interval id must not change

If stored in state → infinite re-renders and bugs ❌ If stored in ref → stable and safe ✅

Key Points (Remember)

The useRef hook in React is used for:

- Accessing DOM elements directly
- Persisting values between renders
- Storing mutable values without causing re-render