React Lecture - 17

useRef Hook

Introduction

When you want a component to “remember” some information, but you don’t want that information to trigger new renders, you can use a ref.

Steps to use useRef hook

1.
```
import { useRef } from 'react';
```

Inside your component, call the useRef Hook and pass the initial value that you want to reference as the only argument. For example, here is a ref to the value o

2.
```
const ref = useRef(0);
```

useRef returns an object like this:

```
{
  current: 0 // The value you passed to useRef
}
```

Steps to use useRef hook

The ref points to a number, but, like state, you could point to anything: a string, an object, or even a function. Unlike state, ref is a plain JavaScript object with the current property that you can read and modify.

Note that the component doesn't re-render with every ref.current update. Like state, refs are retained by React between re-renders. **However, setting state re-renders a component. Changing a ref does not!**

Building a Stopwatch

```
export default function Stopwatch() {
  const [startTime, setStartTime] = useState(null);
  const [now, setNow] = useState(null);
  const intervalRef = useRef(null);

  function handleStart() {
    setStartTime(Date.now());
    setNow(Date.now());

    clearInterval(intervalRef.current);
    intervalRef.current = setInterval(() => {
      setNow(Date.now());
    }, 10);
  }

  function handleStop() {
    clearInterval(intervalRef.current);
  }

  let secondsPassed = 0;
  if (startTime != null && now != null) {
    secondsPassed = (now - startTime) / 1000;
  }

  return (
    <>
      <h1>Time passed: {secondsPassed.toFixed(3)}</h1>
      <button onClick={handleStart}>
        Start
      </button>
      <button onClick={handleStop}>
        Stop
      </button>
    </>
```

Difference between refs and states

refs	state
`useRef(initialValue)` returns `{ current: initialValue }`	`useState(initialValue)` returns the current value of a state variable and a state setter function (`[value, setValue]`)
Doesn't trigger re-render when you change it.	Triggers re-render when you change it.
Mutable—you can modify and update `current` 's value outside of the rendering process.	"Immutable"—you must use the state setting function to modify state variables to queue a re-render.
You shouldn't read (or write) the `current` value during rendering.	You can read state at any time. However, each render has its own snapshot of state which does not change.

When to use refs ?

Typically, you will use a ref when your component needs to “step outside” React and communicate with external APIs—often a browser API that won’t impact the appearance of the component. Here are a few of these rare situations:

- Storing timeout IDs
- Storing and manipulating DOM elements, which we cover on the next page
- Storing other objects that aren’t necessary to calculate the JSX.

If your component needs to store some value, but it doesn’t impact the rendering logic, choose refs.

Limitations of React state don’t apply to refs. For example, state acts like a snapshot for every render and doesn’t update synchronously. But when you mutate the current value of a ref, it changes immediately:

Refs and the DOM

You can point a ref to any value. However, the most common use case for a ref is to access a DOM element.

For example, this is handy if you want to focus an input programmatically.

When you pass a ref to a ref attribute in JSX, like

React will put the corresponding DOM element into myRef.current

Once the element is removed from the DOM, React will update myRef.current to be null