🚀 Day 9 of 60: My React Learning Journey! 🚀

1.What is React Hook ?

React hooks are special functions in React that let us use state and other React features in functional components, which previously could only be done in class components. Hooks provide a simpler way to manage state, handle side effects, and reuse logic across different parts of our application. The most commonly used hooks are \`useState\` for managing state and \`useEffect\` for handling side effects like data fetching or subscriptions. Hooks make it easier to write and maintain functional components by eliminating the need for complex class syntax and lifecycle methods.

2.What is useState hook?

The `useState` hook is one of the fundamental hooks provided by React, enabling functional components to have a state. Before hooks were introduced in React 16.8, state management was only possible within class components. Now, with `useState`, we can manage state in functional components, making them more versatile and powerful

The `useState` hook allows us to add state to functional components. It returns an array containing two elements: the current state value and a function to update that state. This hook can be used multiple times within a single component to manage multiple state variables.

3.How to use useState hook?

Basic Usage:-Here's a simple example to demonstrate how `useState` works.

```
import React, { useState } from 'react';
const Counter = () => {
    // Declare a state variable 'count' and a function 'setCount' to update it
    const [count, setCount] = useState(0);
    return (
        <div>
            <p>Count: {count}</p>
            <button onClick={() => setCount(count +
           1)}>Increment</button>
        </div>
    );
};
export default Counter;
```

In this example:

- `useState(0)` initializes the state variable `count` with a value of `0`.
- `setCount` is a function that updates the state variable `count`.
- Clicking the button increments the count by 1.

Initial State:-The initial state can be any value, including objects, arrays, or even functions. Here's an example with an object.

```
	import React, { useState } from 'react';
	const UserProfile = () => {
		const [user, setUser] = useState({
			name: 'John Doe',
			age: 30,
			job: 'Developer'
		});
		return (
			<div>
				<h1>{user.name}</h1>
				<p>Age: {user.age}</p>
				<p>Job: {user.job}</p>
			</div>
		);
	};
	export default UserProfile;
```

Updating State:-When updating state, it's crucial to understand that calling the state updater function does not merge the new state with the old state. Instead, it replaces the old state. Therefore, when working with objects or arrays, you must explicitly merge the state.

Updating an Object

```
	const updateUser = () => {
		setUser((prevState) => ({
			...prevState,
			age: prevState.age + 1
		}));
	};
```

Updating an Array

```
	const addItem = (newItem) => {
		setItems((prevItems) => [...prevItems, newItem]);
	};
```