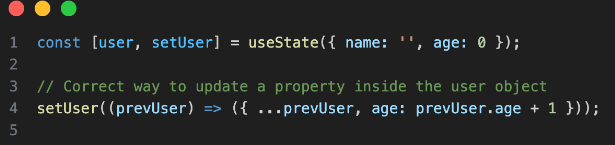
useState()

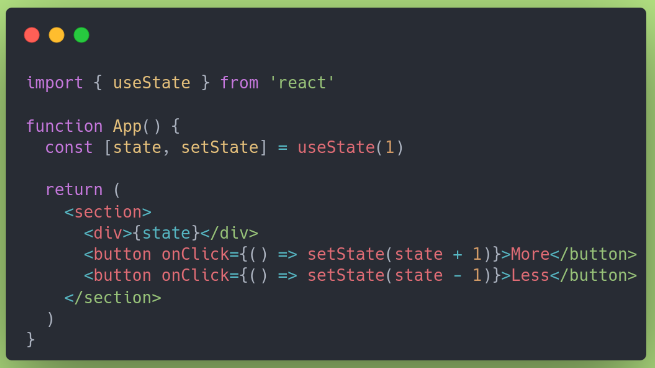
🔗 Connect with Me:  
🌐 LinkedIn : https://www.linkedin.com/in/priya-bagde  
📂 GitHub : https://github.com/priya42bagde  
💻 LeetCode : https://leetcode.com/priya42bagde/  
🎥 YouTube Channel : https://youtube.com/channel/UCK1\_Op30\_pZ1zBs9l3HNyBw (Priya Frontend Vlogz)

* The **state** is **data or properties** which are **mutable**, meaning **their value can change**, using the useState() hook.
* The useState() hook used to add a **state management(**to handle, create, update and manage your states) **in functional components** and **never use it within a nested function, loop or condition**.
* If you have **complex state, then storing multiple values** in useState can get difficult, then the **useReducer hook** which is better suited to managing state with multiple values.
* If the **new value you provide is same to the current state**, then React will **skip re-rendering the component and its children.**
* If you use the **previous value to update state**, you must **pass a function that receives the previous value and returns an updated value**, for example, setMessage(previousVal => previousVal + currentVal)
* The **state updates are asynchronous (i.e, doesn’t** immediately trigger a re-render for each individual state update) and **batched** and **we can use multiple state variables in a component by calling useState hook multiple times**. Also, **React batches multiple setState calls together for a single render**. This means that if you have multiple state updates within the same render cycle, **React will optimize and re-render the component only once**.
* The useState **does not** **automatically merge update objects(**it do shallow merge**)**, so we should **replace the state rather than mutate your existing objects** using **spread operator**. 

**Syntax:** const [count, setCount] = useState(initialState);

* It returns **an array consisting of two elements**: the **current state and a function** to update the state.
* **The first(initial) time** the component is rendered, **the initial state is passed as the argument to useState**. It's **not a mandatory to initiate the state with initial value. it can be a empty useState function**. But initiating the state with initial value will be the part of good practice. In **class components, the state was always an object**, and you could store multiple values in that object. But with hooks, the state can be any type you want as **an array, object, a number, a boolean, a string, whatever you need**.

**Example:**



**Key Point Explanations:**



