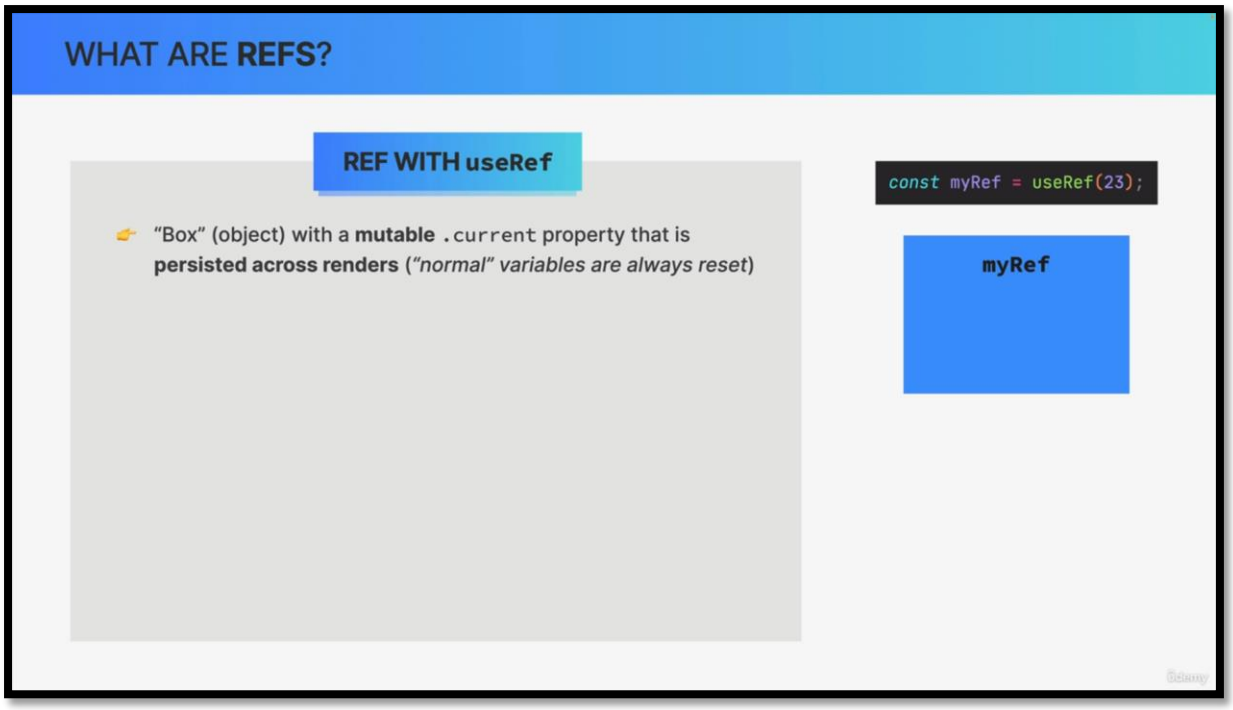
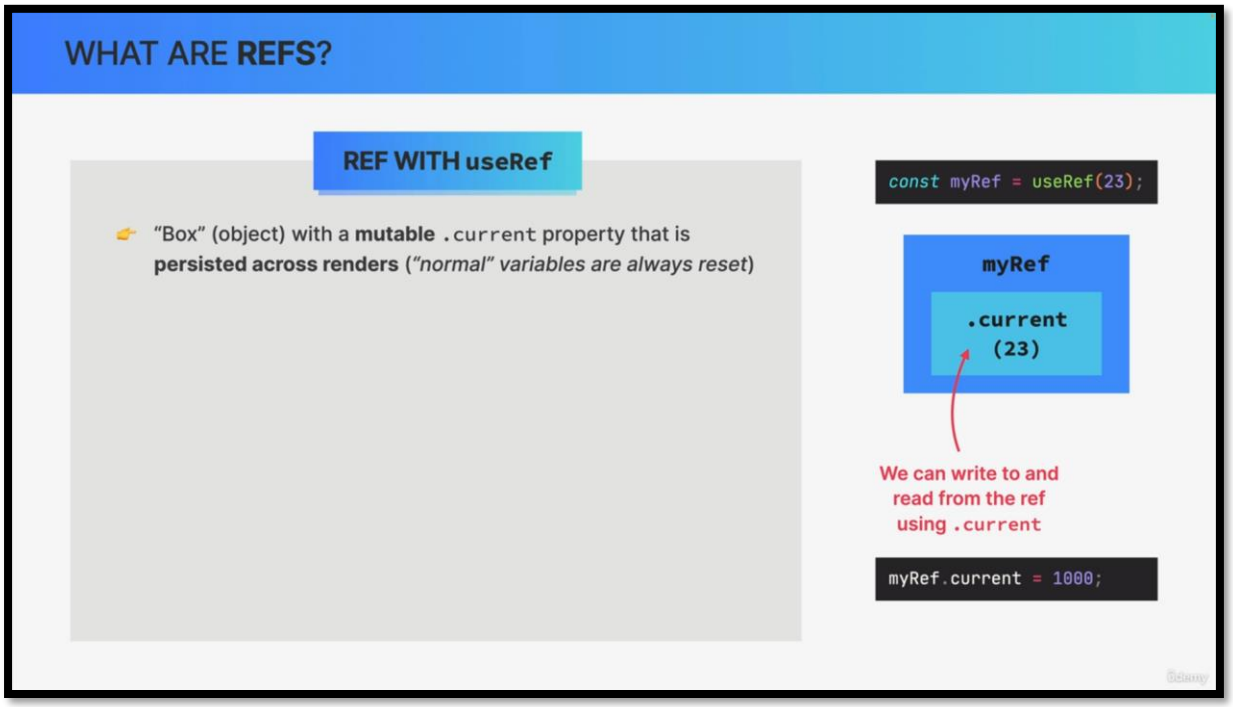


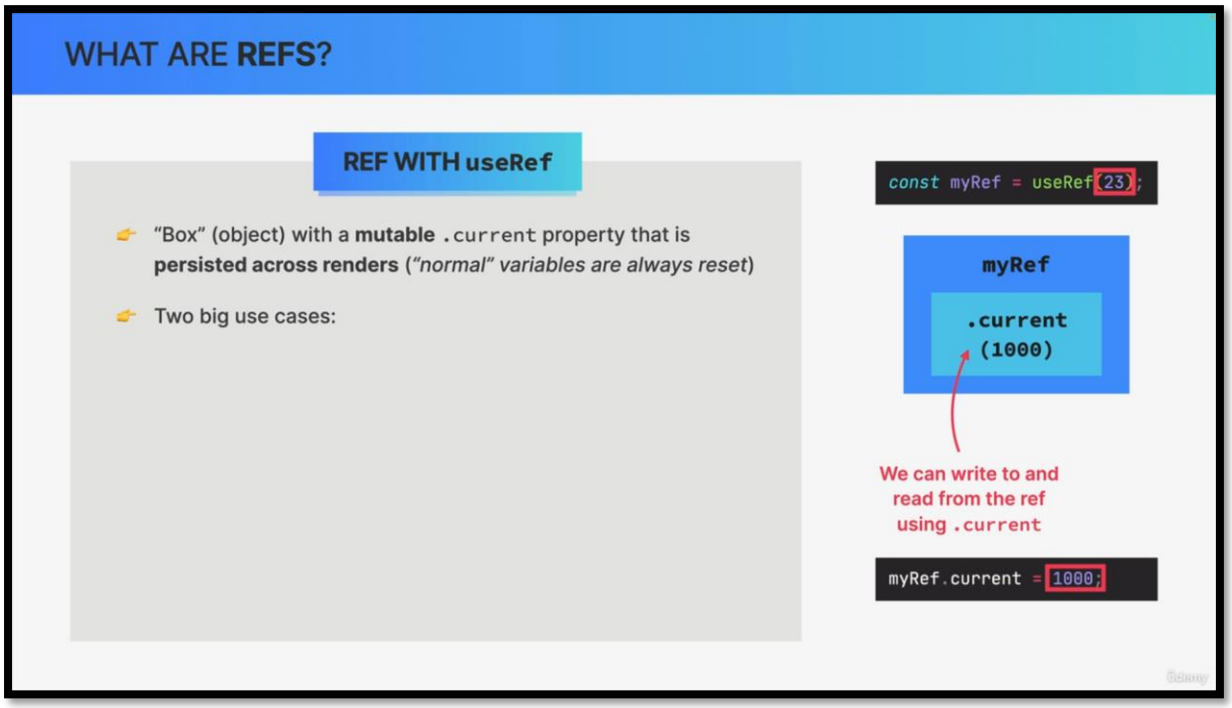
ref stands for reference and it's like a box into which we can put any data that we want to be preserved between renders.



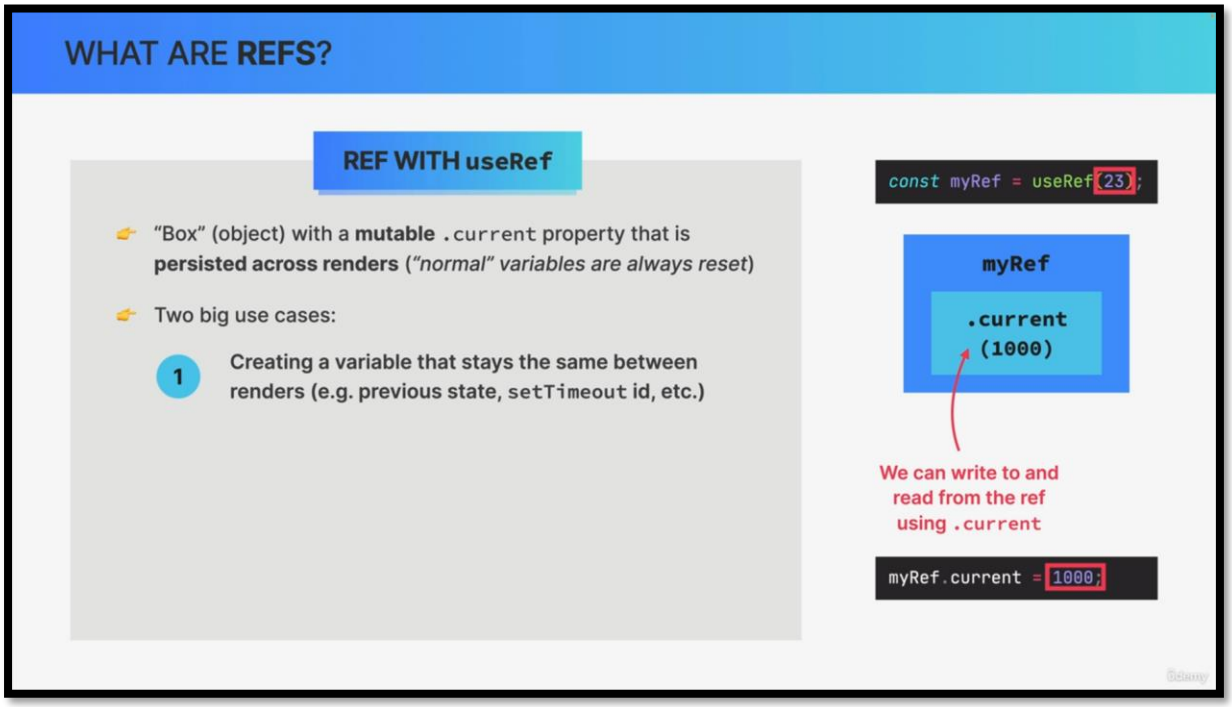
Now, in technical terms, when we use useRef, React will give us an object with a mutable current property and we can then write any data into this current property and, of course, also read from it.

In this small example, the current property was first set to the initial value of 23 and we then changed it to 1000. So, as you can see, this current property is actually mutable, so unlike everything else in React.

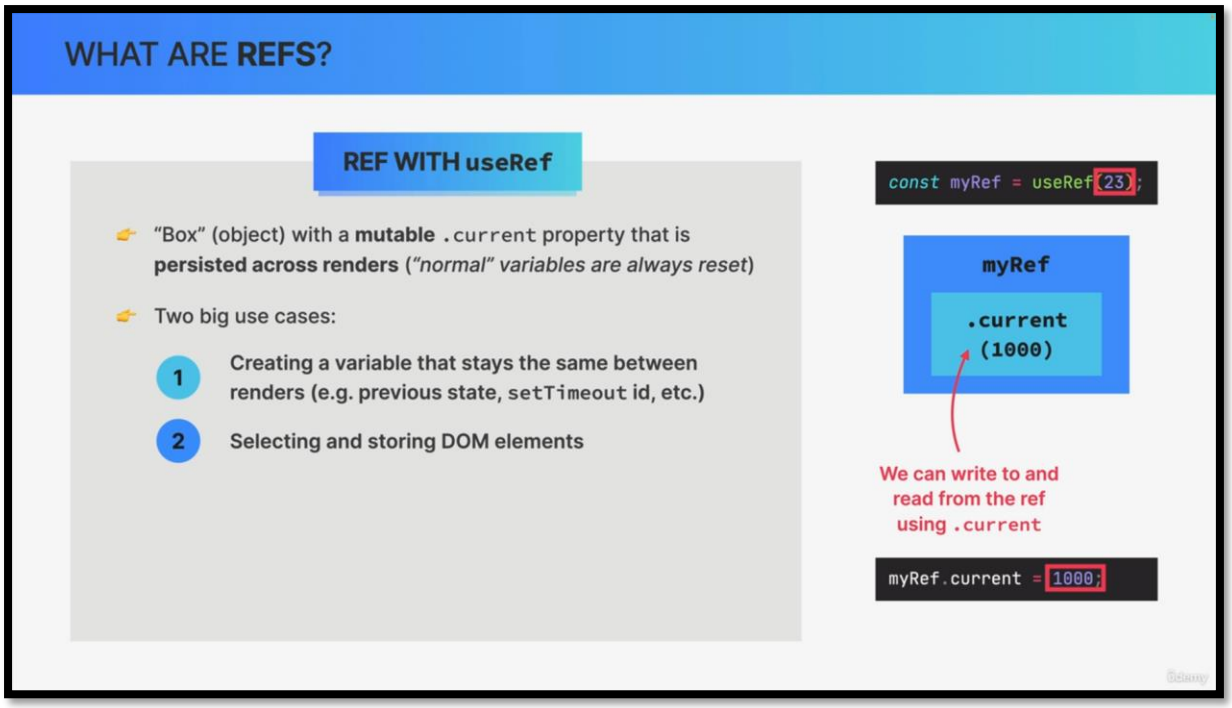
refs are persisted across renders. So, their current property value stays the same between multiple renders, so just like state.



This gives us two big use cases for refs.

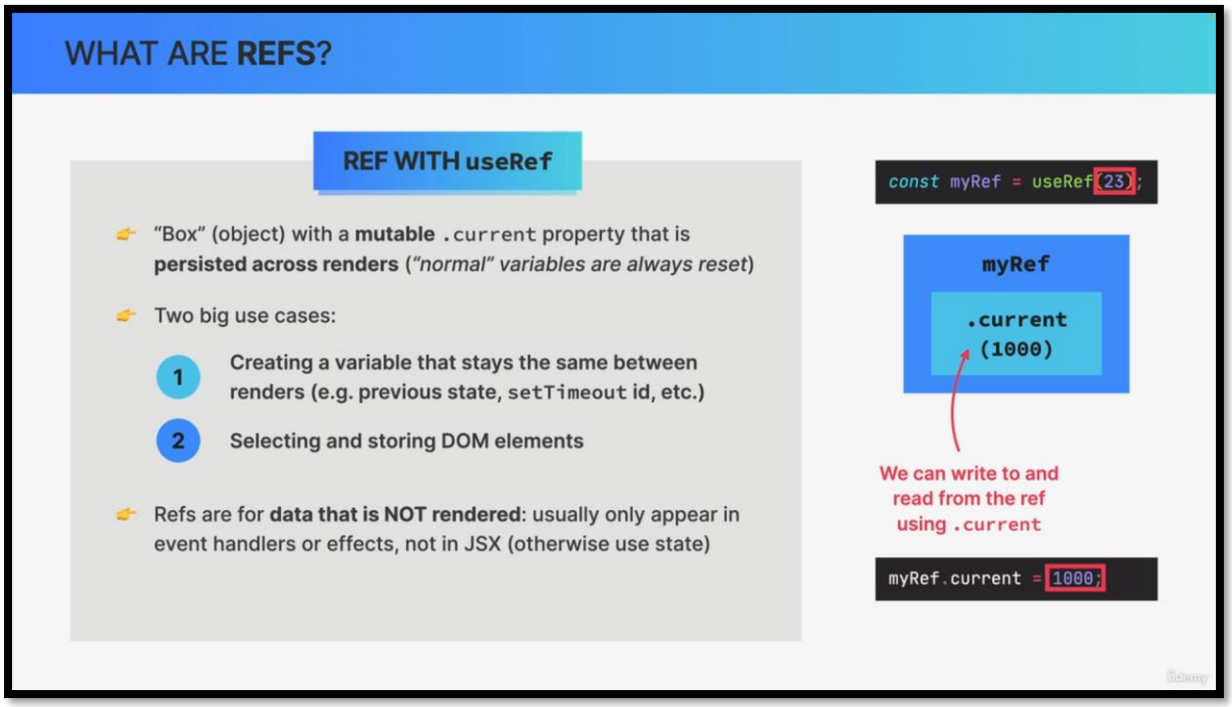


First, as we just said, we can use refs to create variables that will stay the same between renders e.g. preserving the previous state or storing the ID of a setTimeout function.



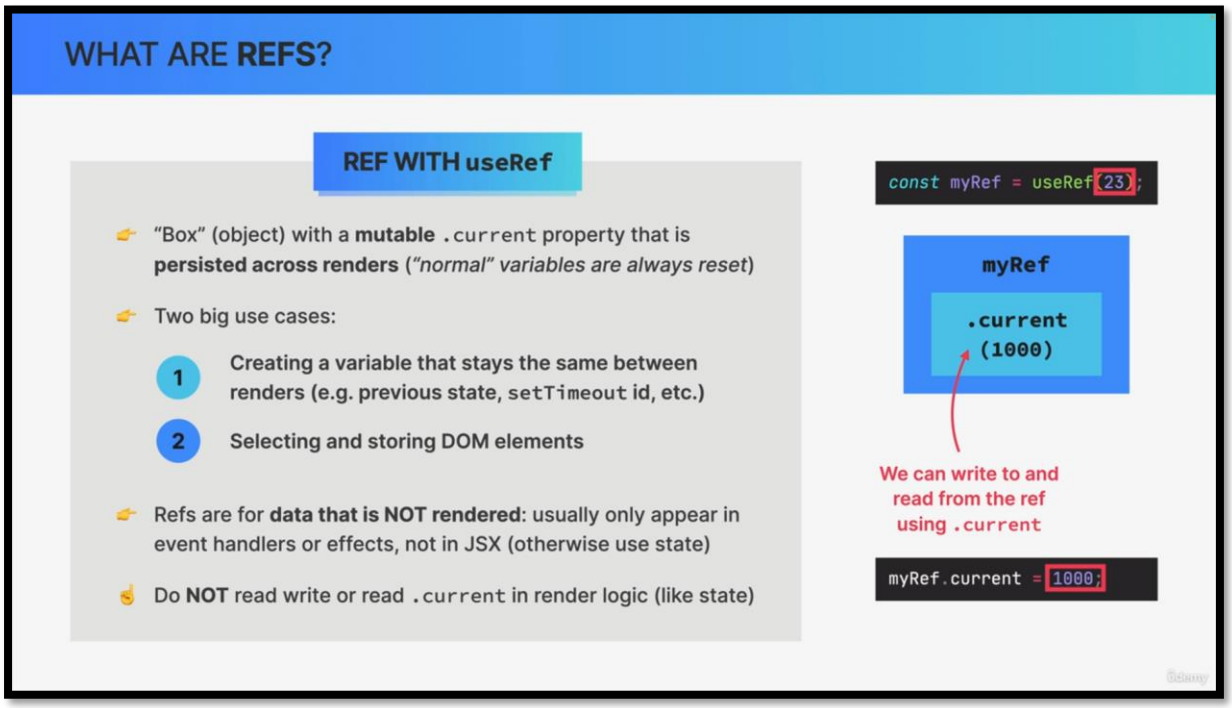
The second use case is actually far more important, which is to select and store DOM elements.

A DOM element is also a piece of data that we want to store and preserve across renders and so refs are perfect for this.



Now refs are usually for data that is not rendered individual output of the component. So, usually refs only appear in event handlers or effects, but not in the JSX.

We can use refs inside JSX too but usually that's not the place for them. If you need data that participates in the visual output of the component, that's usually a good sign that you actually need state and not a ref.



Just like with state, you are not allowed to write or to read the current property in render logic as that would create an undesirable side effect. Instead, we usually perform these mutations inside a useEffect hook.