## Getting Rid of Extra State Variables

Let's clean up the useExpanded.js file by removing extra state variables

```
WE'LL COVER THE FOLLOWING ^
Using a ref Object
The Final Look
Quick Quiz!
```

## Using a ref Object #

As discussed in the last lesson, I prefer to provide the resetDep via a ref object instead of creating a state variable. This way I'm sure to introduce only important variables as actual state variables.

The solution is similar, it's only an internal change within our custom hook, useExpanded. How the user consumes resetDep remains the same.

So, here's the implementation with a ref object:

```
// useExpanded.js
...
const resetRef = useRef(0)
  const reset = useCallback(
    () => {
        // perform actual reset
        setExpanded(initialExpanded)
        // update reset count
        ++resetRef.current
     },
     [initialExpanded]
)
...
// expose resetDep within "value"
...
resetDep: resetRef.current
```

And that's it! Same functionality, different approaches. Feel free to use

whichever feels right to you. I pick the second though:)

As a summary, remember that with the state initializer pattern you offer the user the ability to decide the initial state within your custom hook. You allow for resetting and invoking a custom function after a reset is made as well.

Even if you had a more complex custom hook with multiple state values, you could still use this technique. Perhaps, receive the initial state passed in by the user as an object. Create a memoized <a href="initialState">initialState</a> object from the user's state and use this within your custom hook.

If you also choose to manage your internal state via useReducer, you can still apply this pattern.

The point is, regardless of your implementation, the goal remains the same; to provide the user with the ability to decide the initial state and allow them to reset and invoke custom callbacks after a reset is made.

## The Final Look #

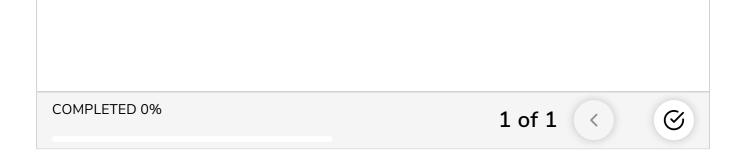
Here is the "Terms and Conditions" expandable object put together so far.

```
.Expandable-panel {
   margin: 0;
   padding: 1em 1.5em;
   border: 1px solid hsl(216, 94%, 94%);;
   min-height: 150px;
}
```

## Quick Quiz! #

Lets take a quick quiz before moving on!

Why have we decided not to keep resetDep as a state variable?



In the next chapter, we'll take things a notch higher as we give even more control to the users of our components. Remember, that's what building highly reusable components is about!