What Is the State Reducer Pattern?

Let's get started with the reducer pattern!

WE'LL COVER THE FOLLOWING

Why the State Reducer Pattern?

The **state reducer pattern** is the final and perhaps the most advanced pattern that we'll be discussing in this course. Don't let that scare you. I'll take my time to explain how it really works while explaining why it matters too.

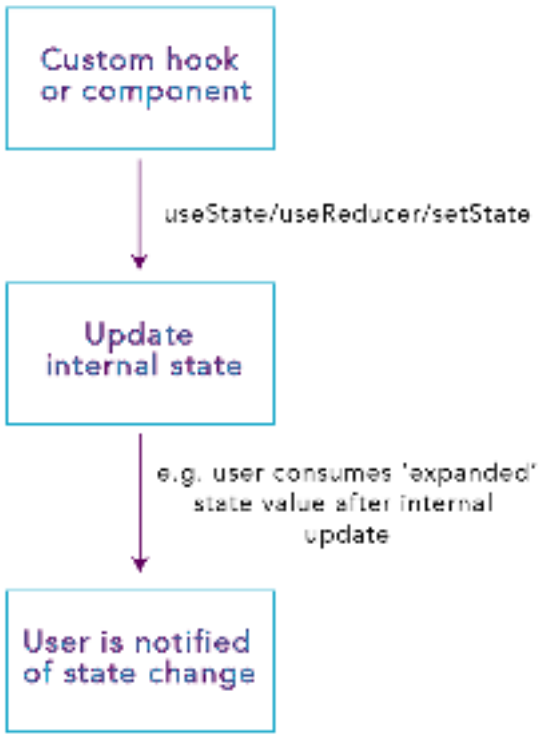
When we write custom hooks or components, they most likely have some way to keep track of the internal state. With the state reducer pattern, we give control to the user of our custom hook/component on how the state is updated internally.

State Updating In Regular Patterns

The technical term for this is called "inversion of control". In layman terms, it means a system that allows the user to control how things work internally within your API.

Let's look at how this works conceptually.

The typical flow for communicating updates to the user of your custom hook or component looks like this:



Communicating updates in custom hooks/components

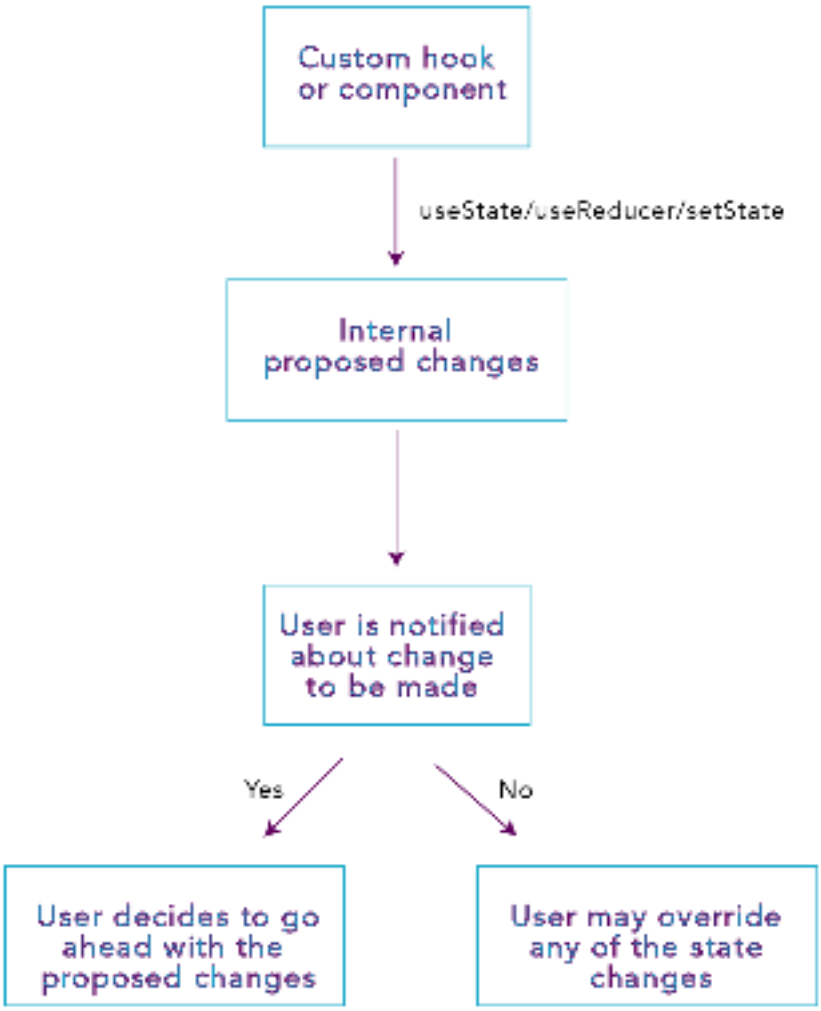
Within your custom hook, you call `useState` or `useReducer` or `setState` to update state internally. Once the update is made, the new state value is communicated to the user.

This is how all the patterns we've looked at so far have worked.

State Updating in the Reducer Pattern

With the state reducer pattern, there's a significant change in how internal state updates are made.

Here's an illustration:



New internal state updation pattern

With state reducers, what's important to note is that *before* we make any internal state update, we send our proposed changes over to the user. If the user is okay with the changes, we go ahead and update state with our proposed changes. If they need to override a state change, they can do that as well.

The internal update is controlled by the user. They have a say!

Quick Quiz!

Let's take a quiz on how the state reducer pattern works.

1 How do updates occur in other patterns?

COMPLETED 0%

1 of 2

Let’s dive right into the implementation-level details of this pattern in the next lesson!