When to choose useReducer vs useState

The `useState` hook is best used on less complex data.

While it's possible to use any kind of a data structure when working with `useState`, it's better to use it with primitive data types, such as strings, numbers, or booleans.

The `useReducer` hook is best used on more complex data, specifically, arrays or objects.

While this rule is simple enough, there are situations where you might be working with a simple object and still decide to use the `useState` hook.

Such a decision might stem from the simple fact that working with `useState` can sometimes feel easier than thinking about how the state is controlled when working with `useReducer`.

It might help conceptualizing this dilemma as a gradual scale, on the left side of which, there is the `useState` hook with primitive data types and simple use cases, such as toggling a variable on or off.

At the end of this spectrum, there is the `useReducer` hook used to control state of large state-holding objects.

There's no clear-cut point on this spectrum, at which point you would decide: "If my state object has three or more properties, I'll use the `useReducer` hook".

Sometimes such a statement might make sense, and other times it might not.

What's important to remember is to keep your code simple to understand, collaborate on, contribute to, and build from.

One negative characteristic of `useState` is that it often gets hard to maintain as the state gets more complex.

On the flip side, a negative characteristic of `useReducer` is that it requires more prep work to begin with. There's more setup involved. However, once this setup is completed, it gets easier to extend the code based on new requirements.

Conclusion

You learned about the decision-making process when choosing between `useReducer` and `useState` for working with different types of data.