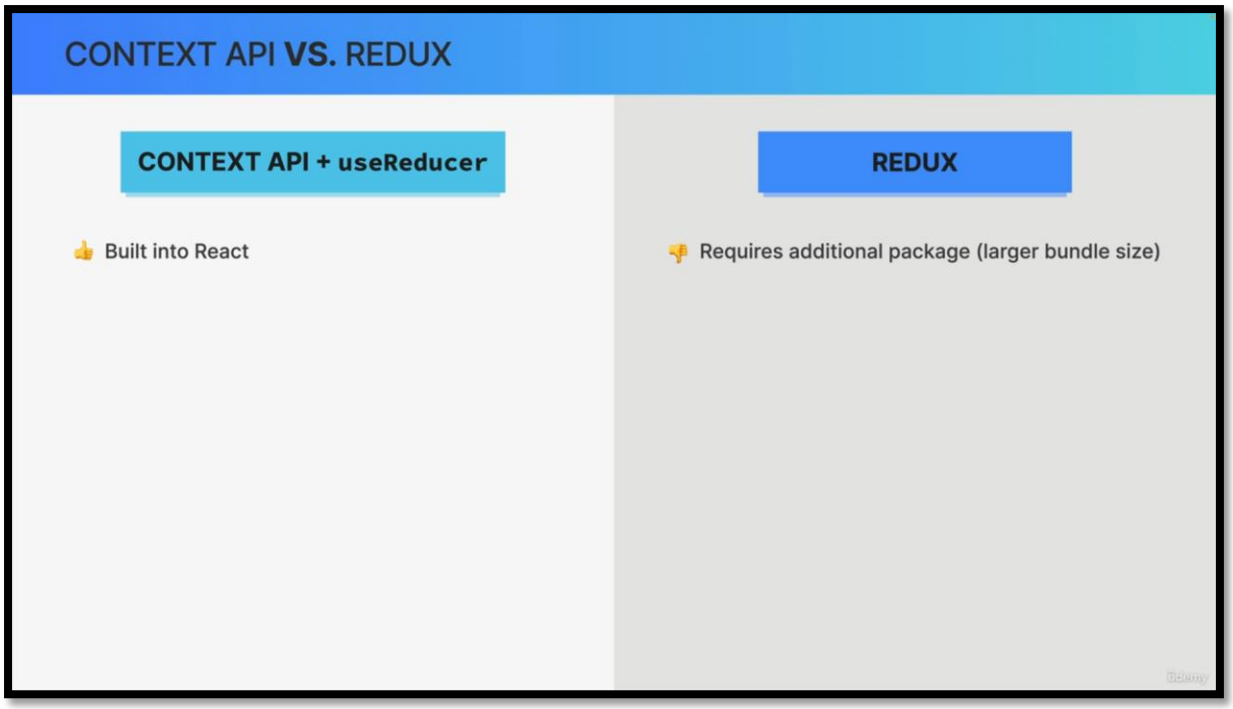
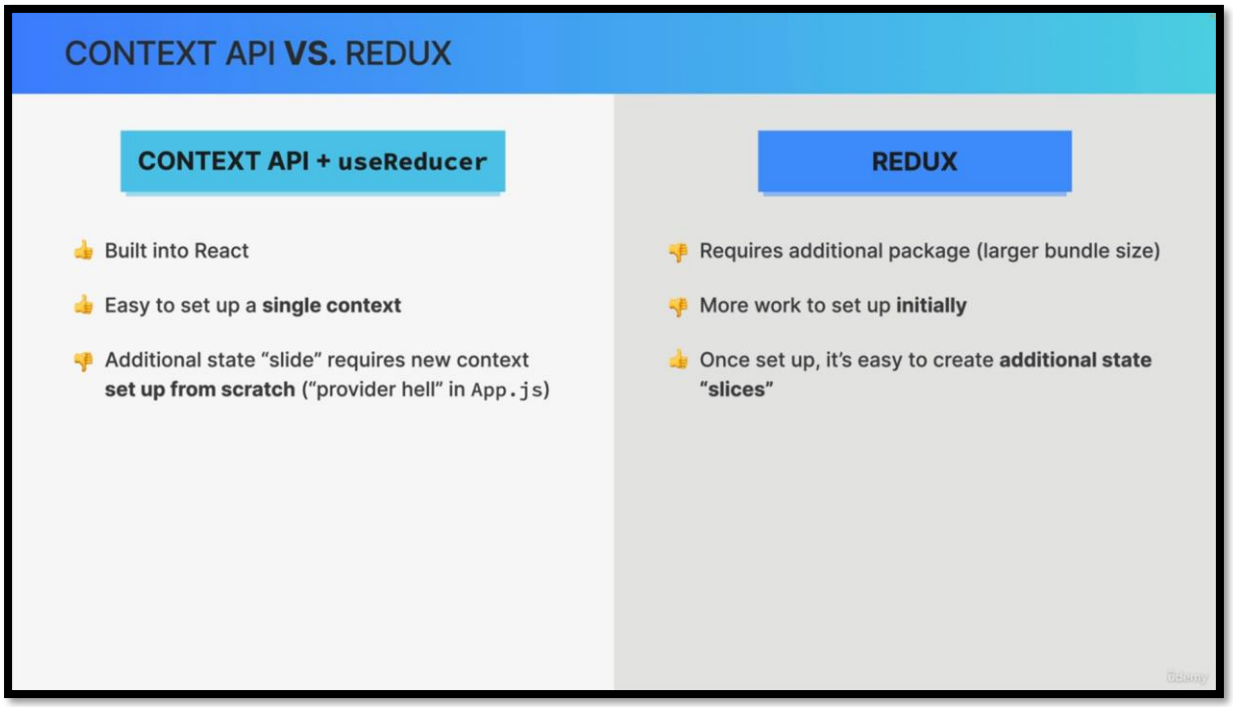


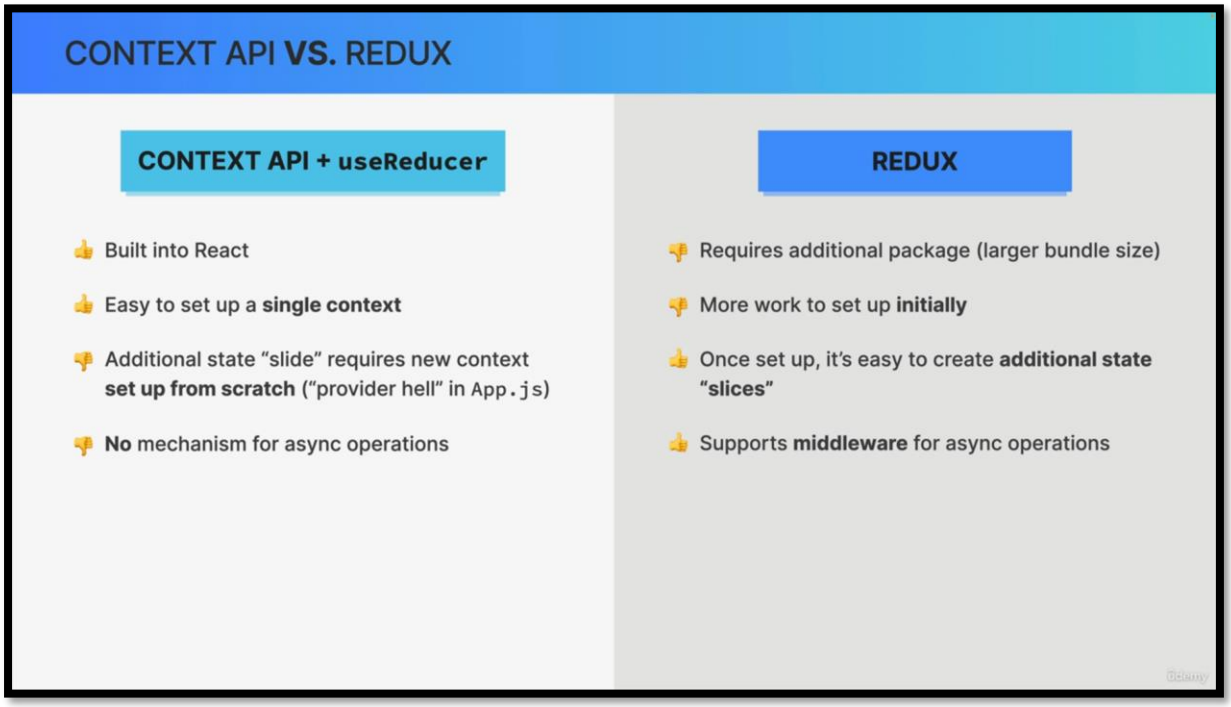
The first advantage of the Context + useReducer solution is that these features are already built into React while if you want to use Redux, you need to install additional packages, which is not only more work, but will also increase your bundle size.



Now in terms of setup, it's quite straightforward to set up a single context and useReducer. Pass the state value into the context, and then provide it to the app. However, for each additional state slice, you will need to do the same thing again.

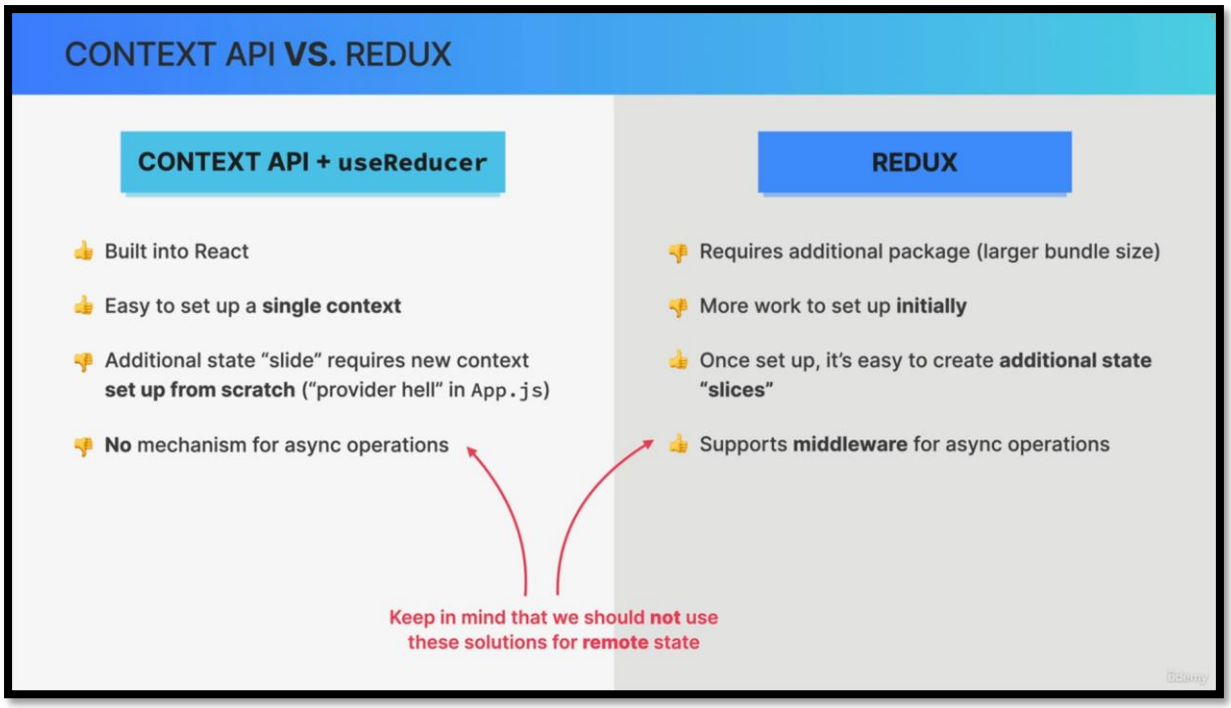
So, you will need to set up everything from scratch for each new piece of state, or each new slice of state, and this can lead to something that we call "provider hell", where you end up with many, many context providers in your main app component.

On the other hand, Redux requires a bit more work to do the initial setup, but once that is done, it's quite straightforward to add additional state slices. All you need to do, is to create a new slice file and add your reducers, but you don't need to create another provider and another custom hook.

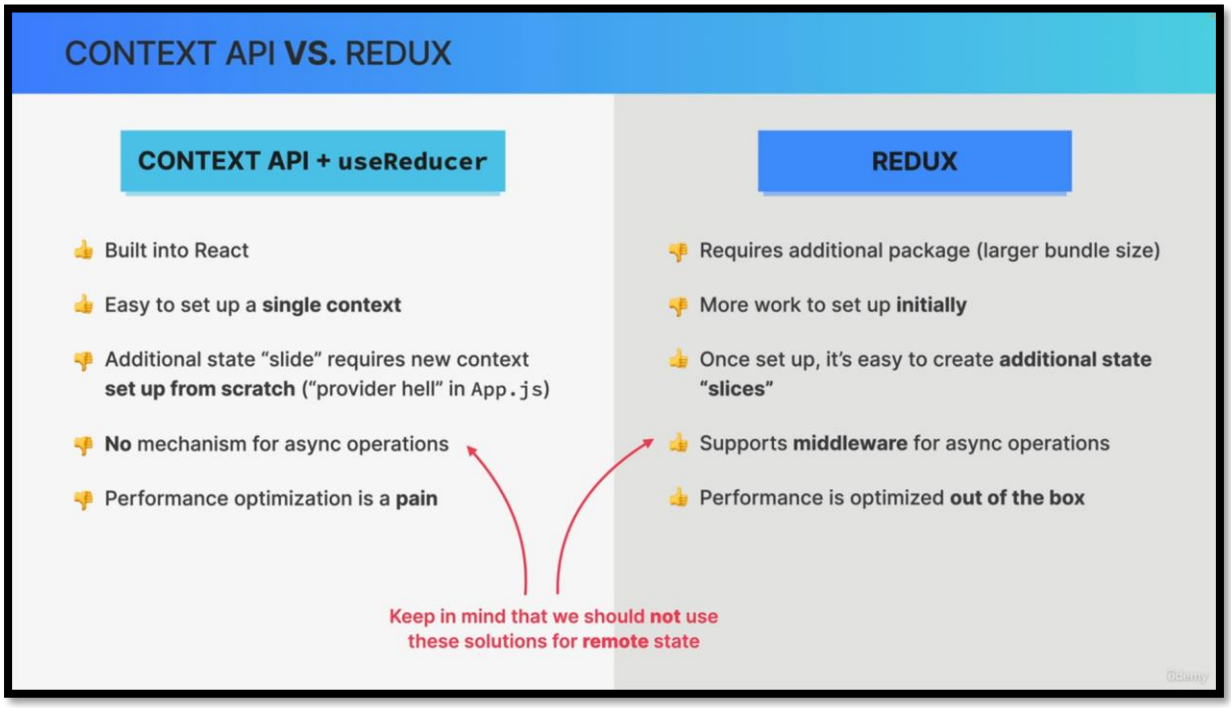


Now, when it comes to async operations, like fetching data, the context API has no built-in mechanism for that, as we have experienced ourselves, in the worldwide application.

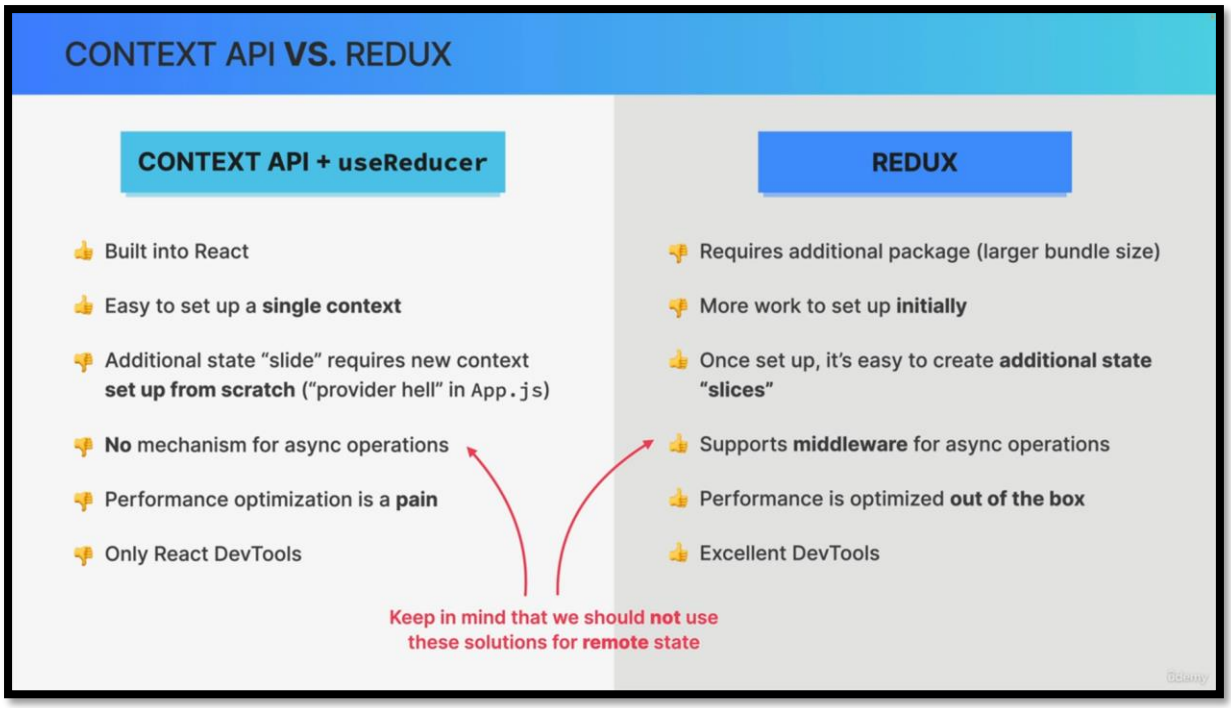
Redux, on the other hand, has support for middleware, and while middleware is not necessarily easy to use, it does give us a way to handle asynchronous tasks right inside the state management tool.



Just keep in mind that we should probably not use any of these tools for remote state anyway, but sometimes it can still be useful to be able to fetch just some single data point from an API. So, just like we did with the currency conversion.



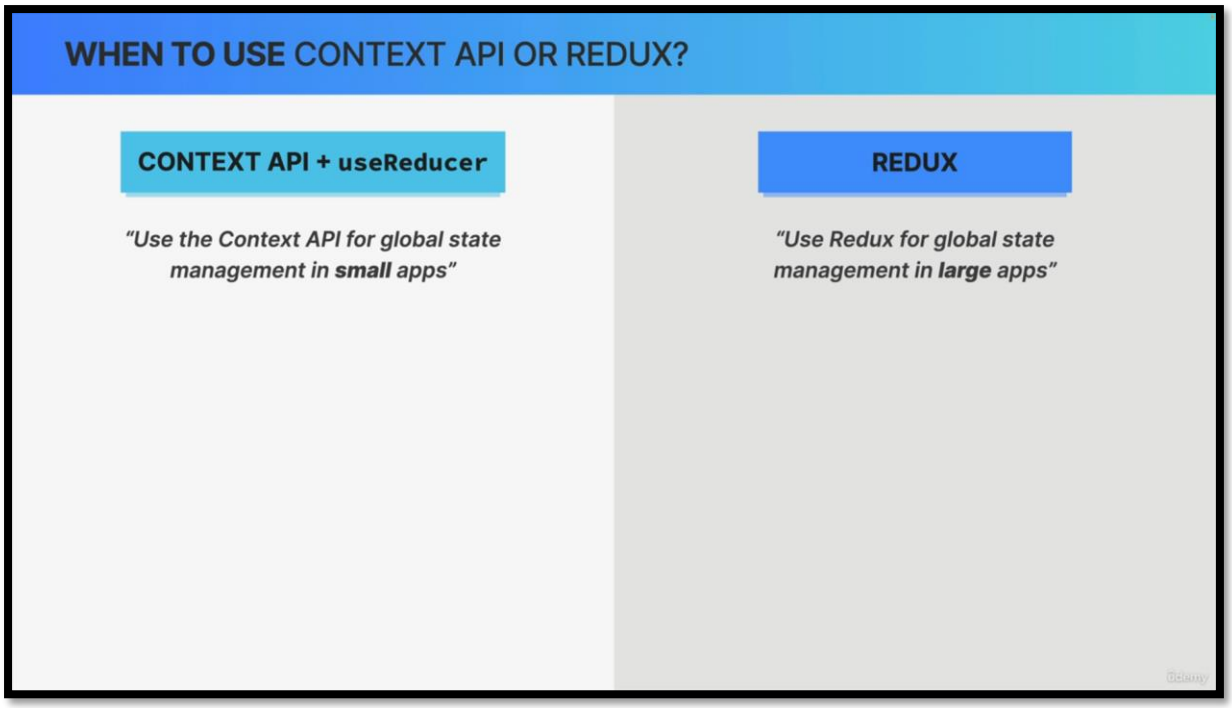
Optimizing the performance of the context API and useReducer solution can require some work, while Redux comes with many optimizations out of the box. So, including minimizing wasted renderers.



Finally, to finish our list of pros and cons, let's consider that Redux has some excellent DevTools. While for the context API, we can only use the simple React developer tools, and this can make a huge difference in applications where the state is truly huge, and complex, and is updated a lot.

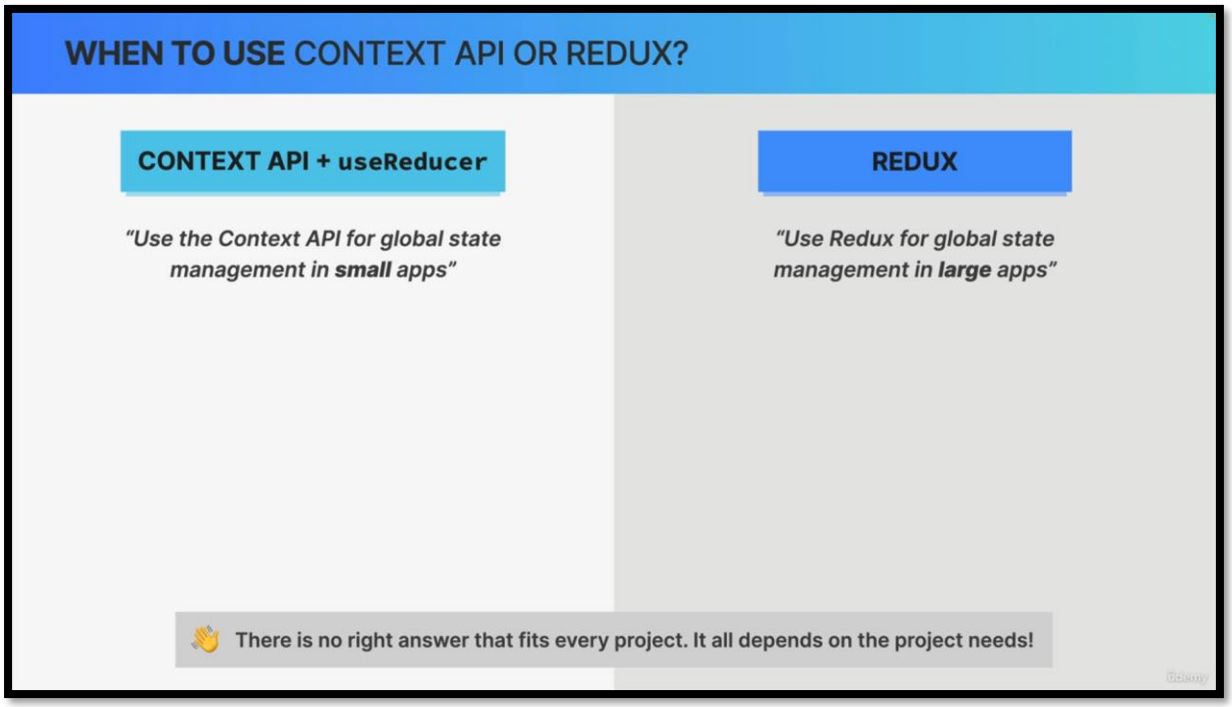
So, from this list, it appears that Redux has way more pros than the context API, but that's not really the point here.

So, we're not counting the pros and cons but now based on these facts, let's move on to some actual usage recommendations.



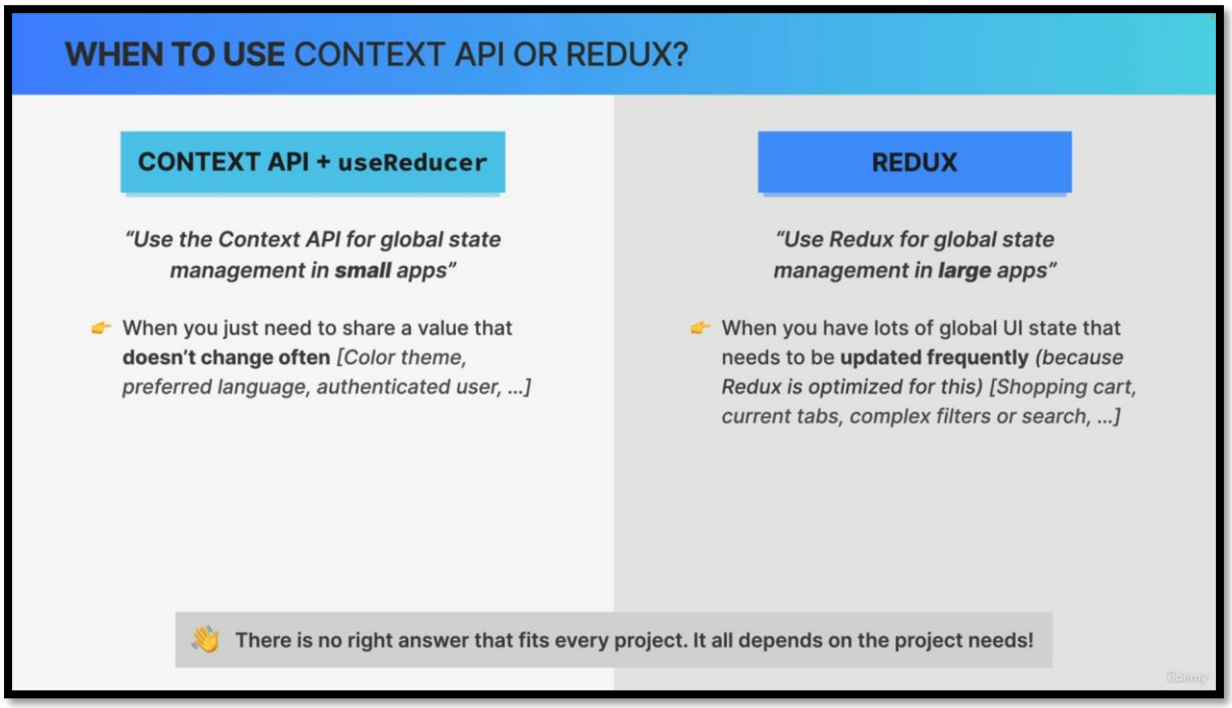
So, the general consensus seems to be use the context API plus React hooks for global state management in small applications and use Redux to manage global state in large apps.

But that's not super helpful, and so let's dig a bit deeper.



Just know that, in general, there is never a right answer that is gonna fit every single project out there.

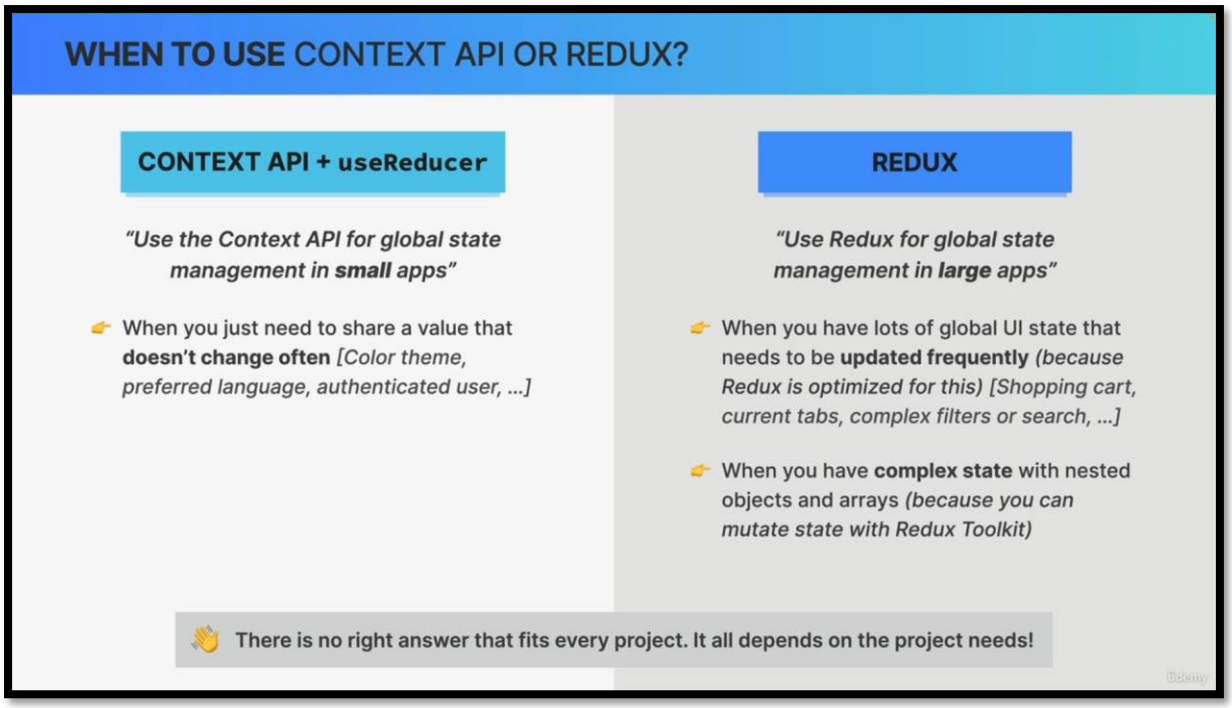
So, the technical choice between any of these solutions will always depend entirely on the project's needs.



But with that being said, when, for example, all you need is to share a value that doesn't change very often, then the context API is perfect for that and some examples are the app color theme, the user's preferred language, or the currently authenticated user.

So, these will rarely change, and so you can use a context for those because there will be no need to optimize the context in this situation.

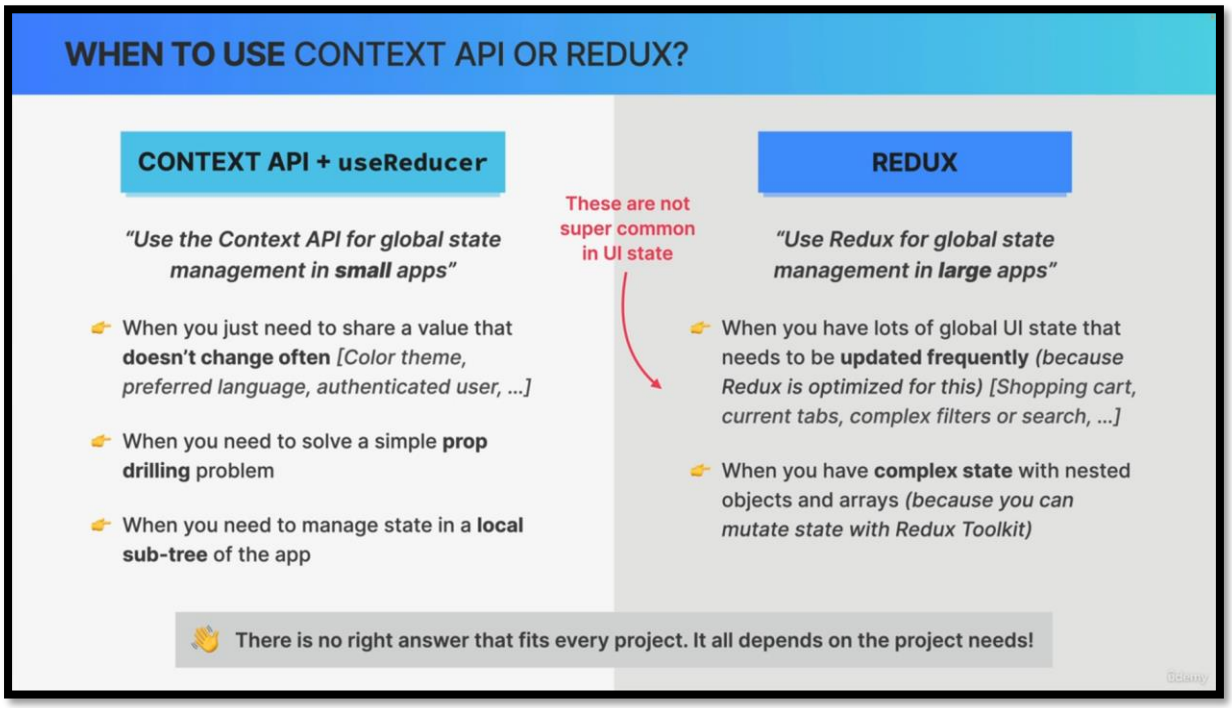
Now on the other hand, when you do have lots of UI state that needs frequent updates, like shopping carts, the currently open tabs, or some complex data filters, then Redux might be the way to go and this is because as we just learned, Redux is already heavily optimized for these frequent updates.



As we also already touched on, Redux is perfect when you have complex state with nested objects, because then you can mutate state in Redux Toolkit, which is really, really helpful.

So, these are the two situations in which I would personally reach for Redux.

Now, they're not that common when it comes to UI state, and so, again, this is why Redux has fallen a bit out of favor recently.



Now, going back to the context API, it can be very helpful when we have a simple prop drilling problem that we need to solve, or when we need to manage some state in a local sub-tree of the application.

So, in that case, it's not really global state, but state that is global to a smaller sub-part of the app, basically.