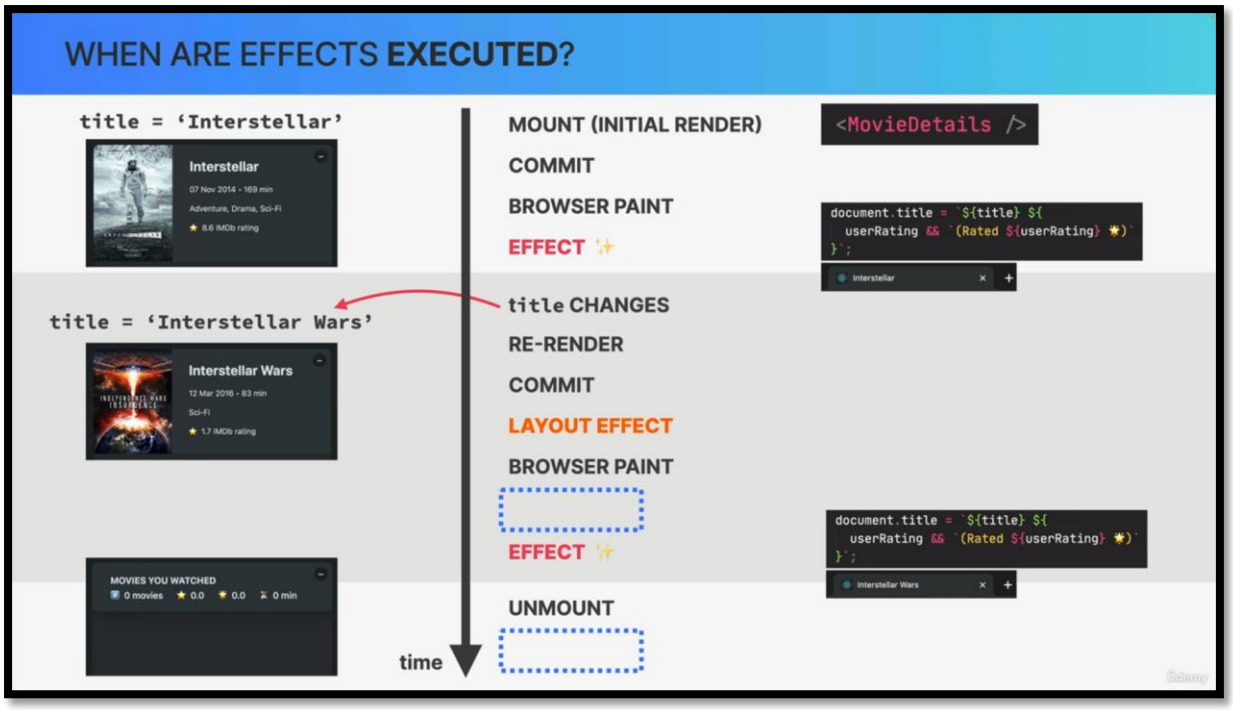
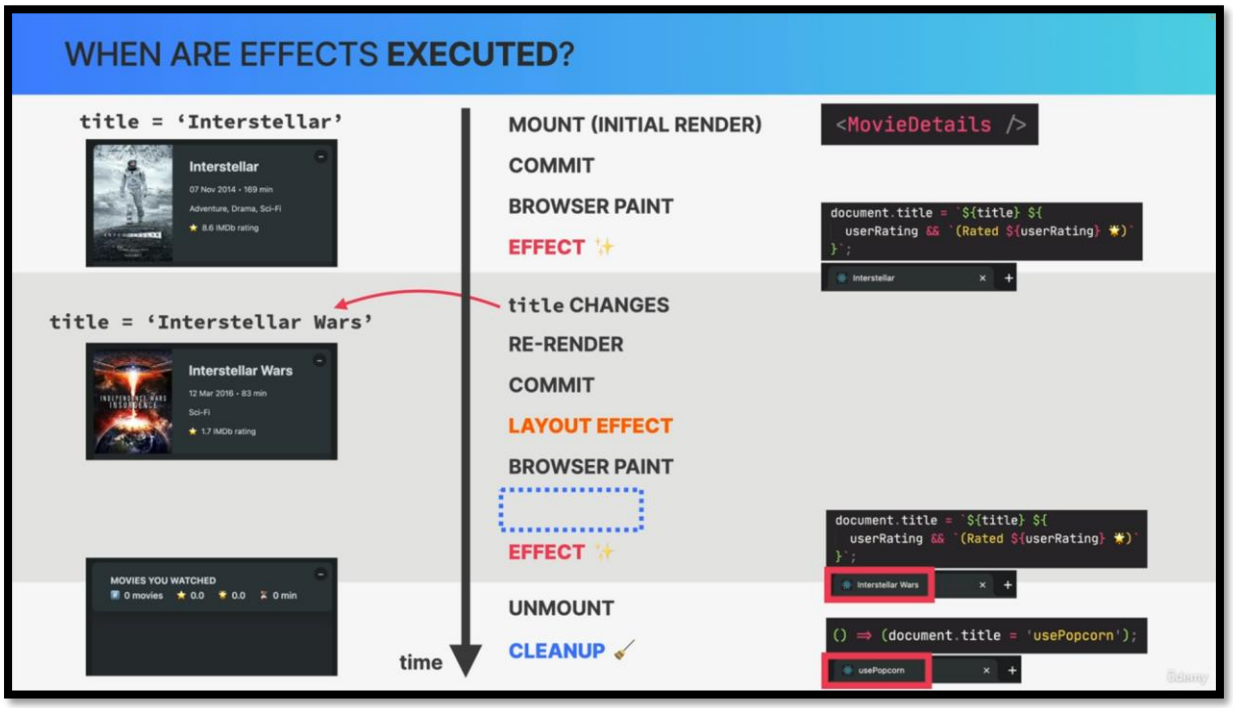


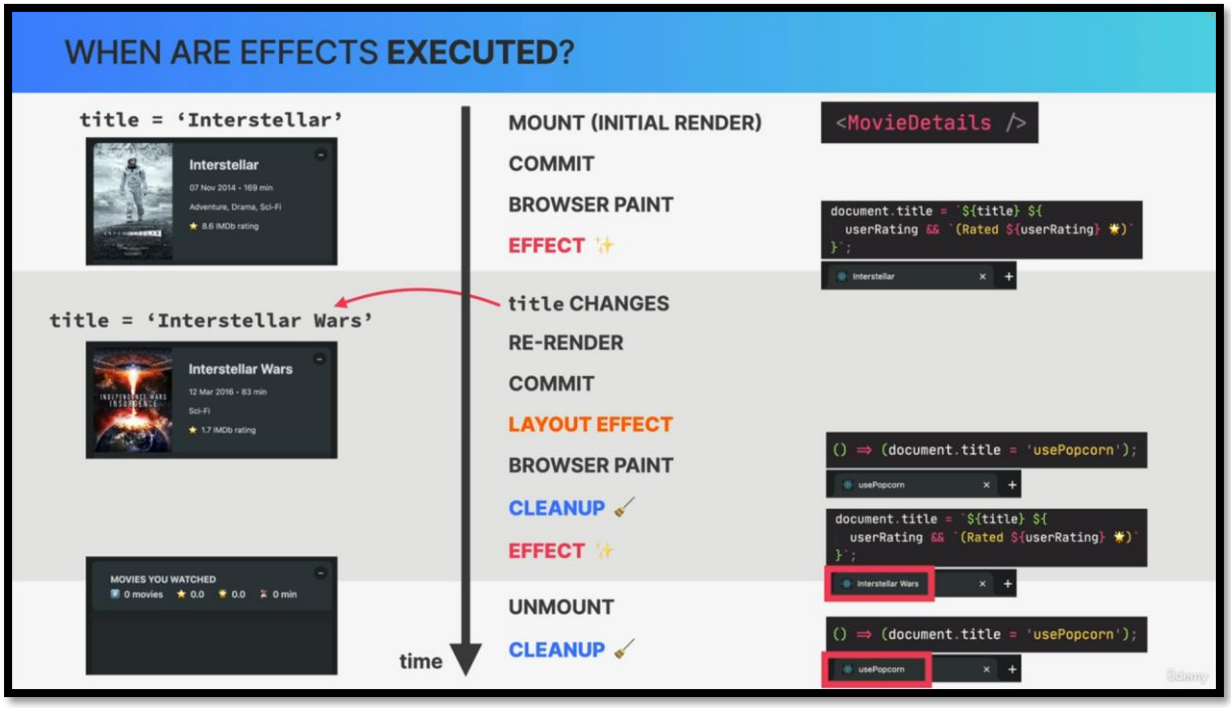
After the last effect run, the title of the page in the browser tab was set to Interstellar Wars.

However, once we unmounted the movie details component, we would probably like the title to return to the original text, which was simply usePopcorn, so just the name of the application.

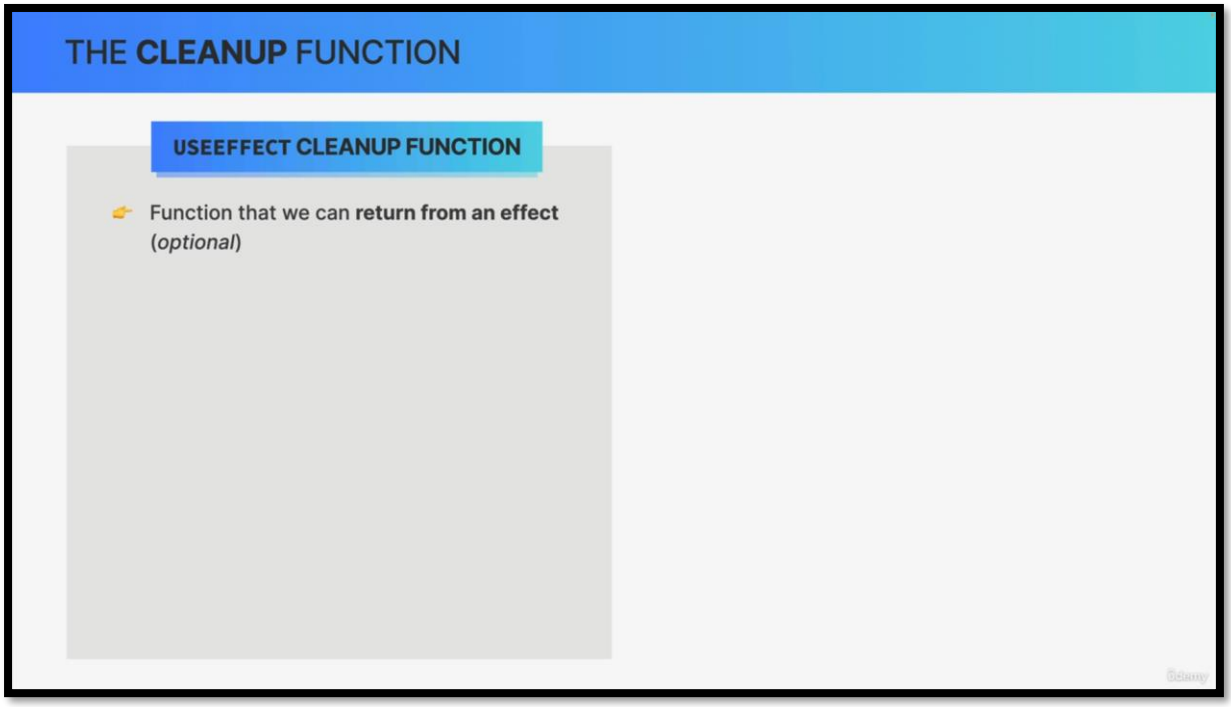
How can we ensure that the page title stays synchronized with the application, even after the component has disappeared?



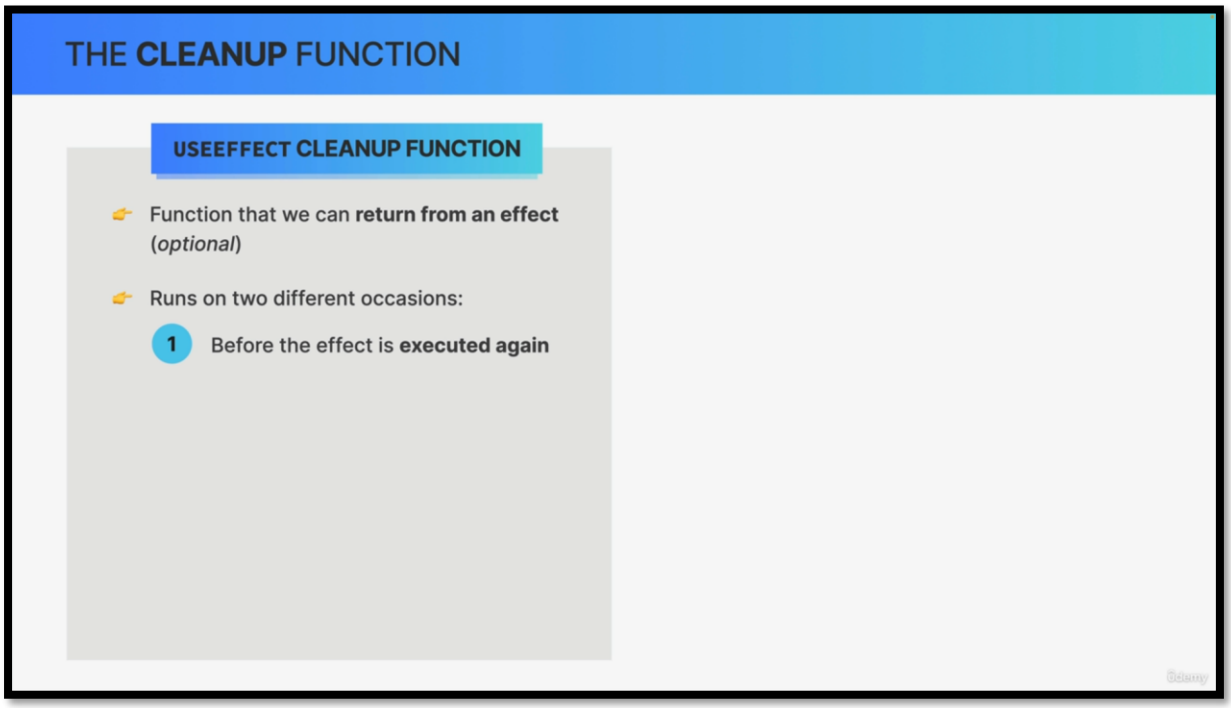
Well, basically what we need is a way to execute some code as the component unmounts. We can do exactly that by returning a so-called cleanup function from the effect. In this case, that's simply a function that sets the title back to usePopcorn.



We still have another hole here in the timeline, and that's because the cleanup function that we return from the effect is actually also executed on re-renders, so right before the next effect is executed again.

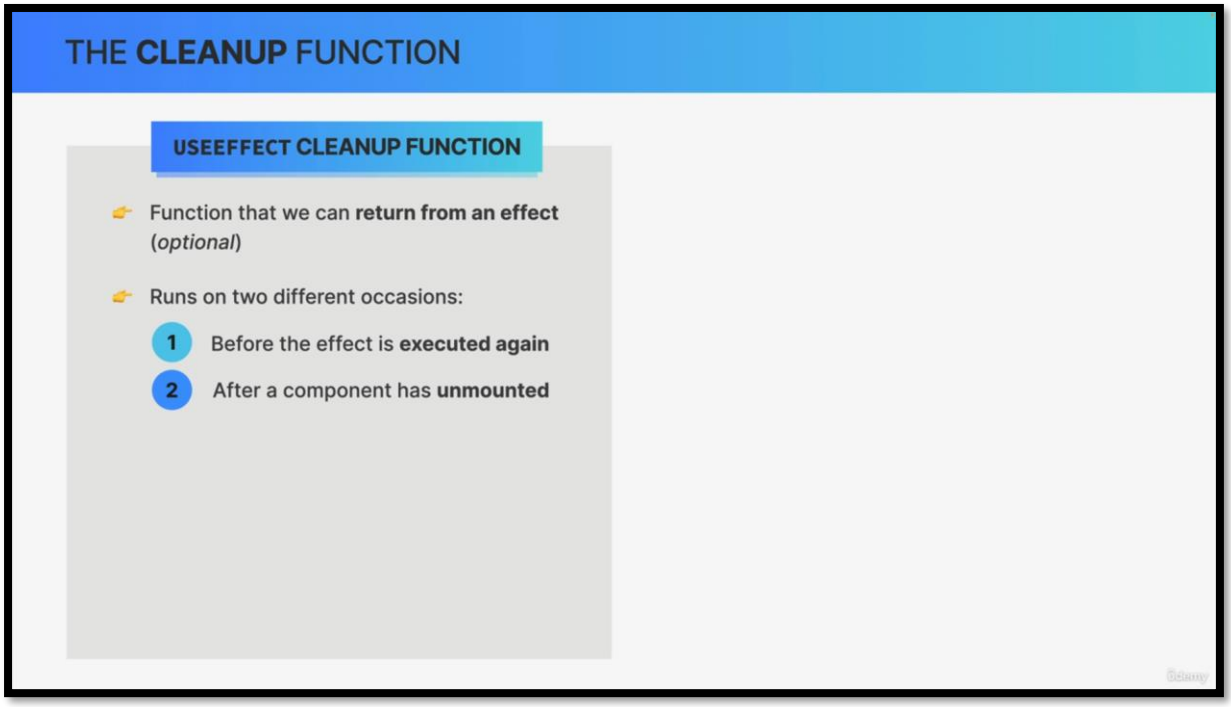


The cleanup function is a function that we can return from an effect. The cleanup function is optional, so we don't have to return one from the effect.

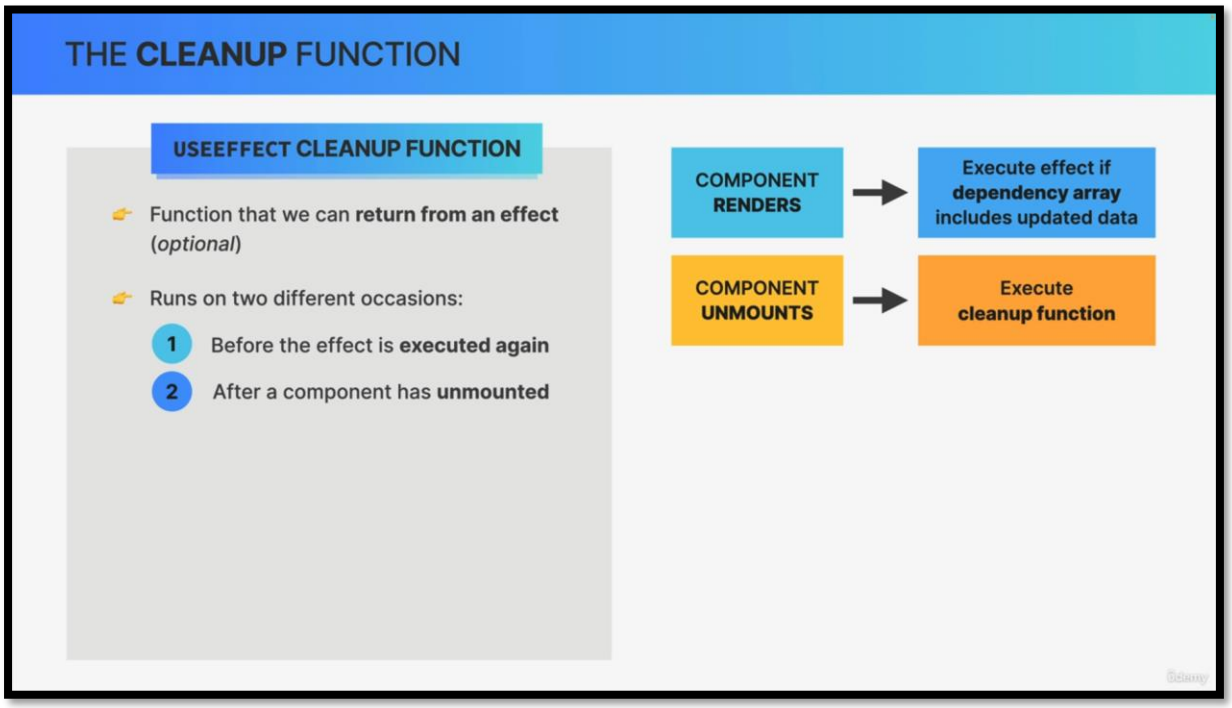


Now the cleanup function will run on two occasions.

First, it runs before the effect is executed again, in order to clean up the results of the previous side effect.



It also runs right after the component instance has unmounted, in order to give us the opportunity to reset the side effect that we created, if that's necessary.

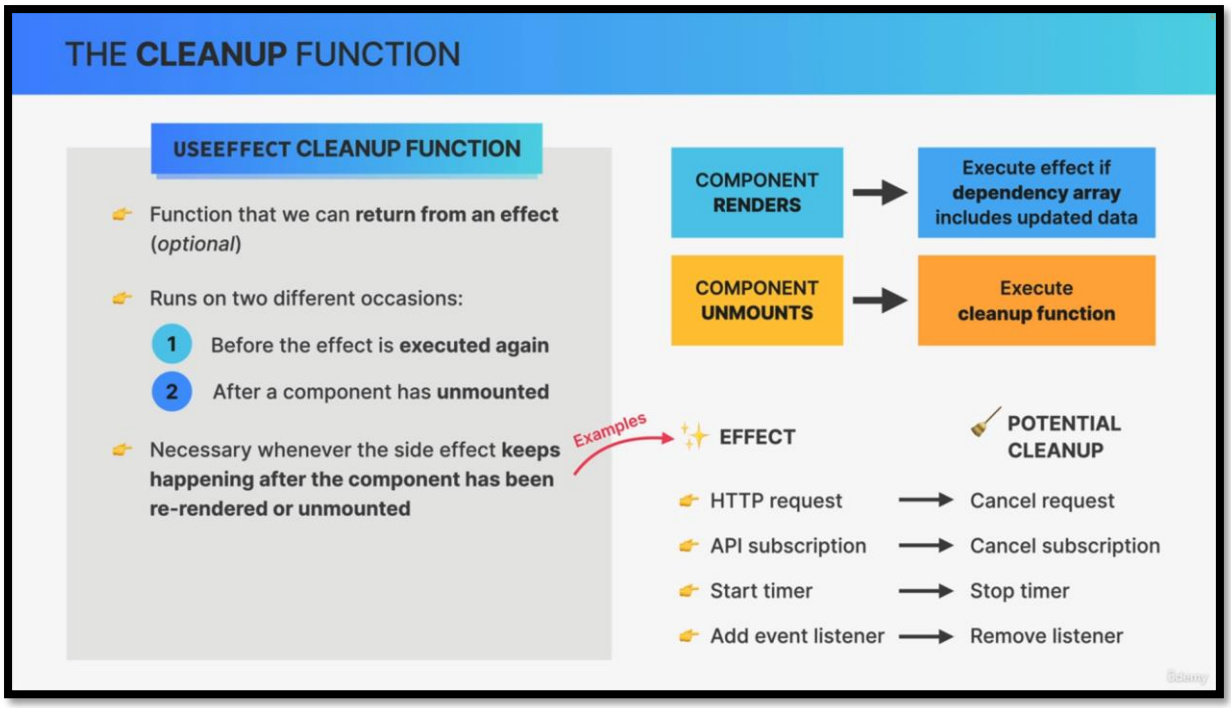


We have the dependency array, in order to run code whenever the component mounts or re-renders. With the cleanup function, we also have a way to run some code whenever the component unmounts.

With this, we have the entire component life cycle covered.

When do we actually need a cleanup function?

Well, basically we need a cleanup function whenever the side effect keeps happening after the component has been re-rendered or unmounted.



For example, you might be doing an HTTP request in your effect.

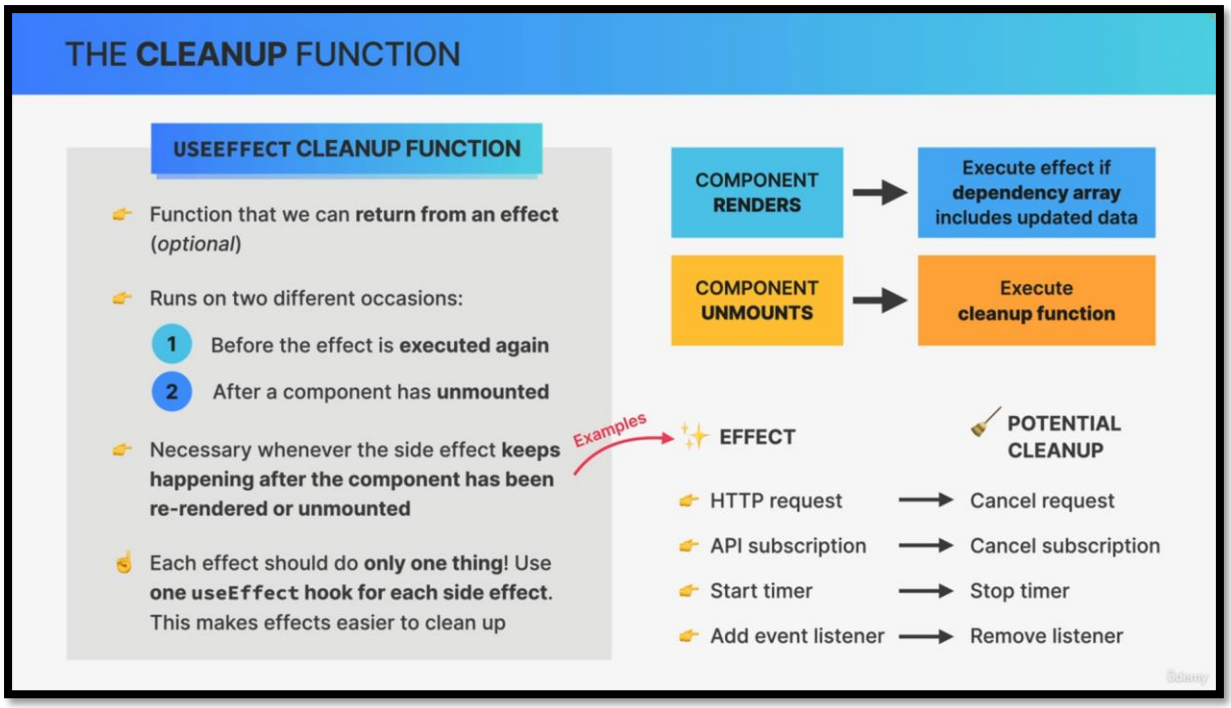
Now if the component is re-rendered while the first request is still running, a new second request would be fired off. So, this might then create a bug called a race condition. Therefore it's a good idea to cancel the request in a cleanup function whenever the component re-renders or unmounts.

There are many other examples.

So, when you subscribe to some API service, you should cancel the subscription.

When you start a timer, you should stop the timer in the cleanup function.

If you add an event listener, you should clean up by removing it.



Each effect should only do one thing. So, if you need to create multiple effects in your components, which is completely normal, just use multiple useEffect hooks. This not only makes each effect much easier to understand, but it also makes effects easier to clean up using a cleanup function.