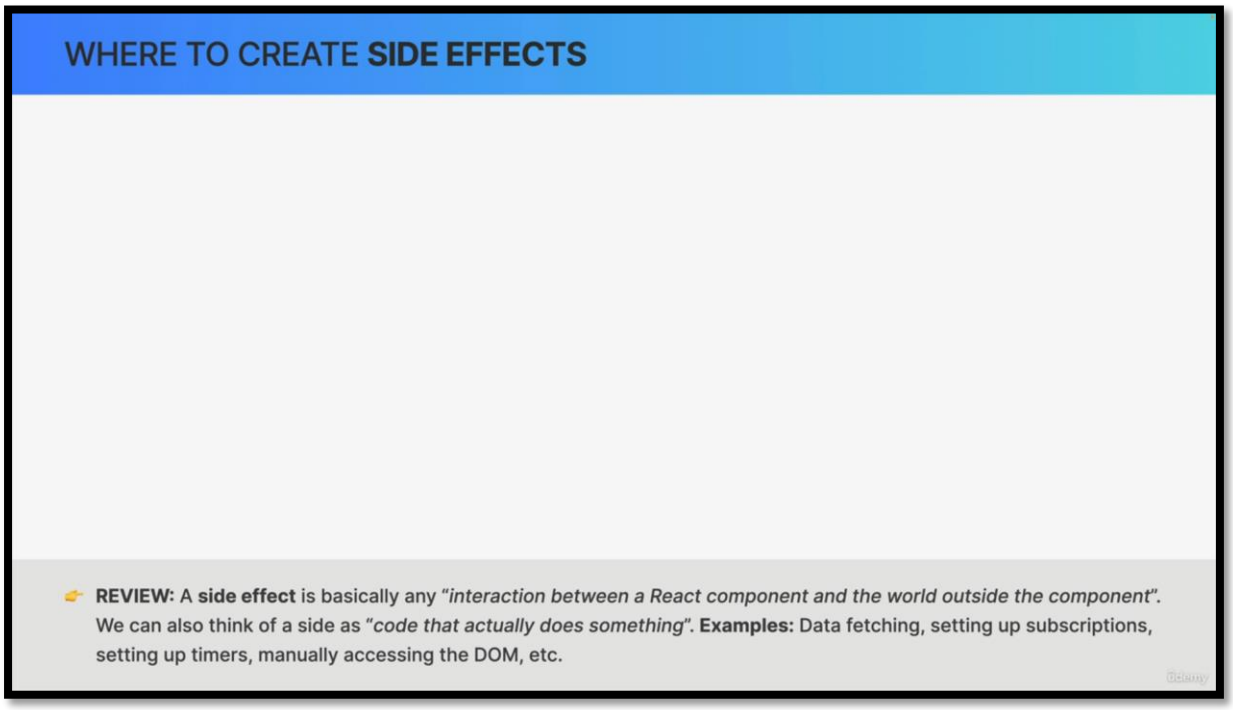
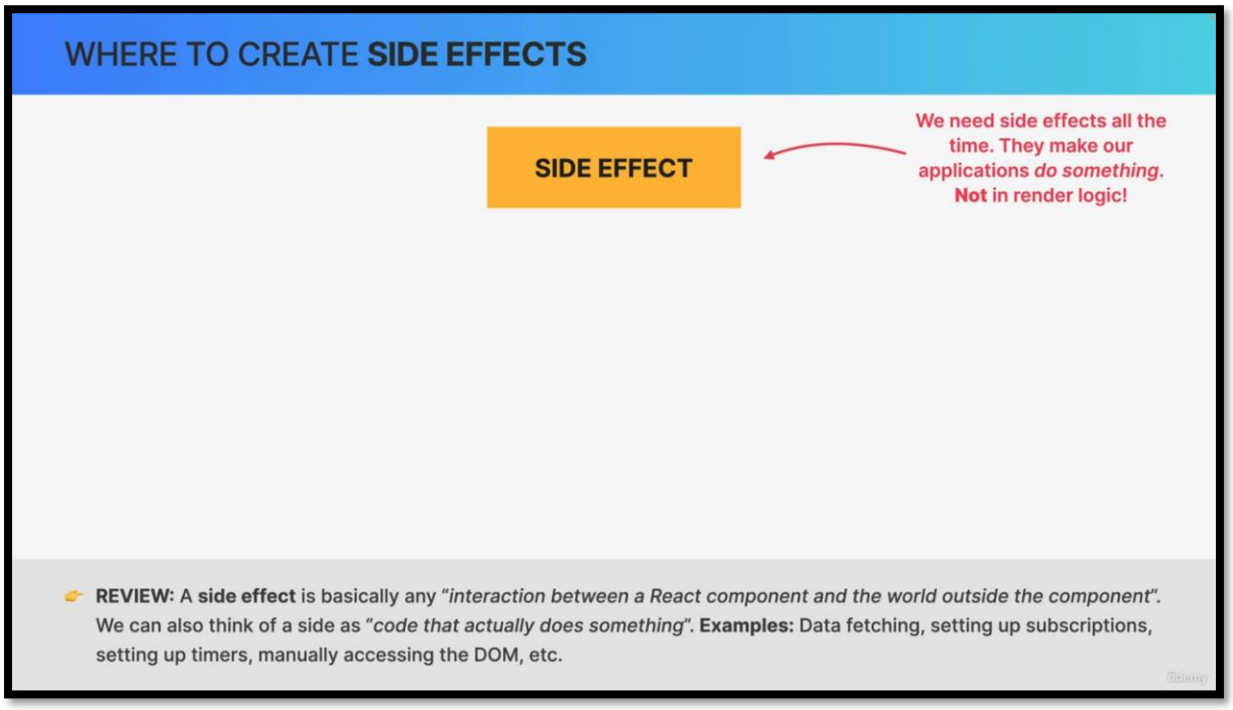
The idea useEffect hook is to give us a place where we can safely write side-effects. The side-effects registered with useEffect hook will only be executed after certain renders.

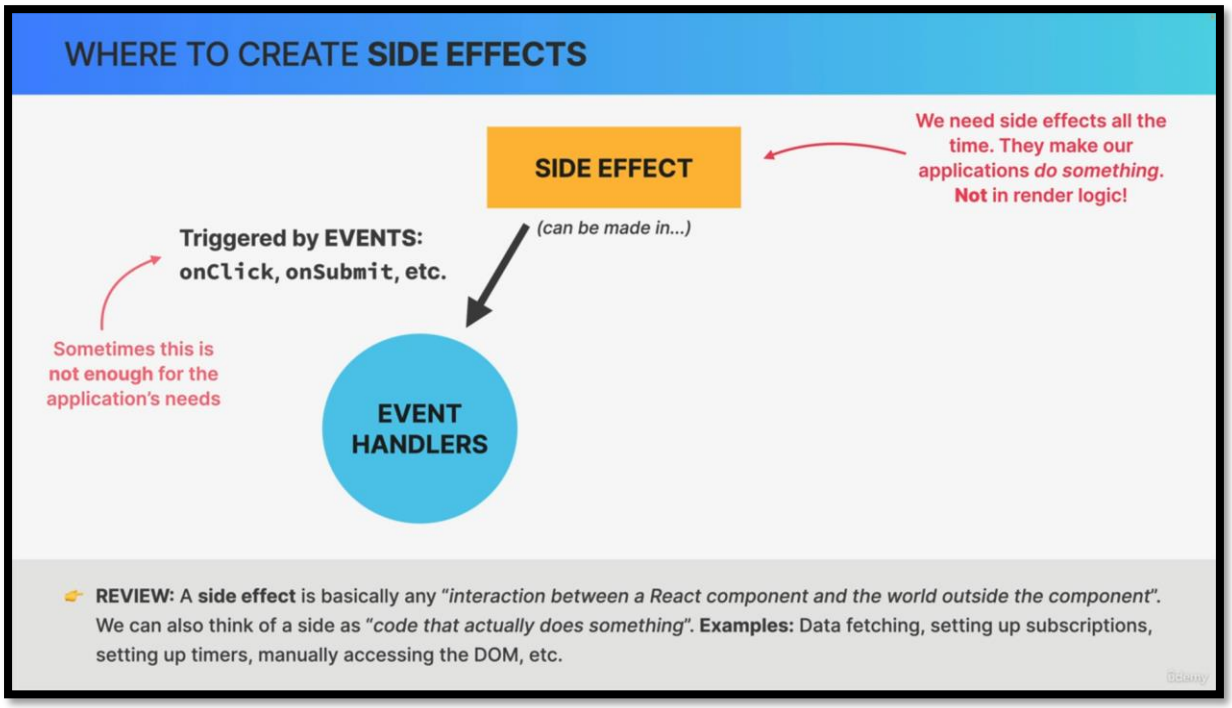


In React, a side effect is any interaction between a React component and a world outside that component.



We can think of a side effect as some code that actually makes something useful happen. For example, fetching data from some API. So, what this

means is that we actually need side effects all the time when we build React apps.

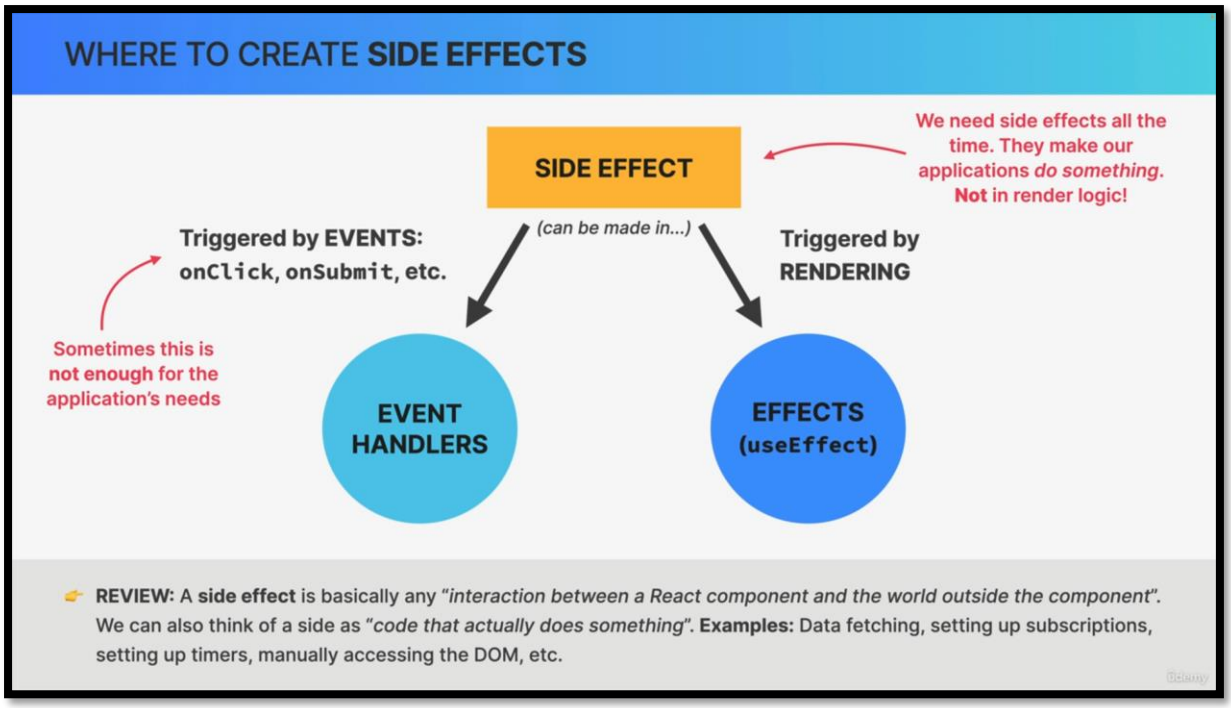


Side effects should not happen during the component render, or in other words side effects should not be in render logic.

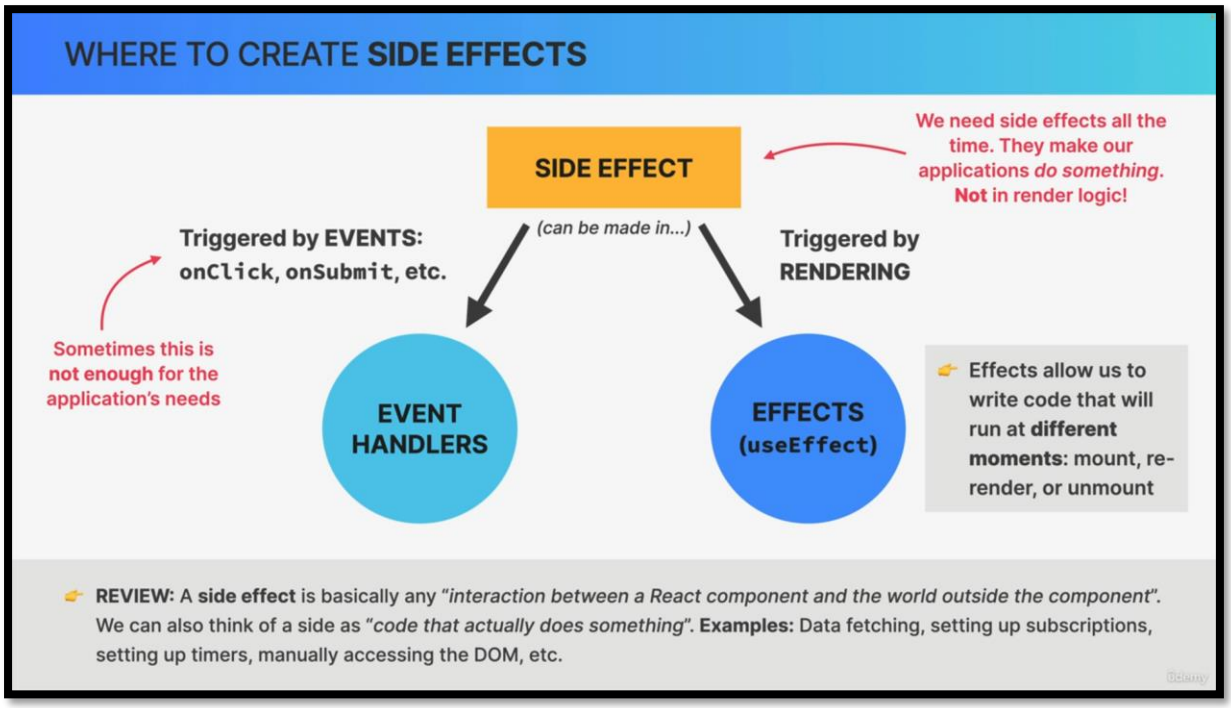
We can create side effects in two different places in React.

The first one is inside event handlers. Event handlers are simply functions that are triggered whenever the event that they are listening to happens.

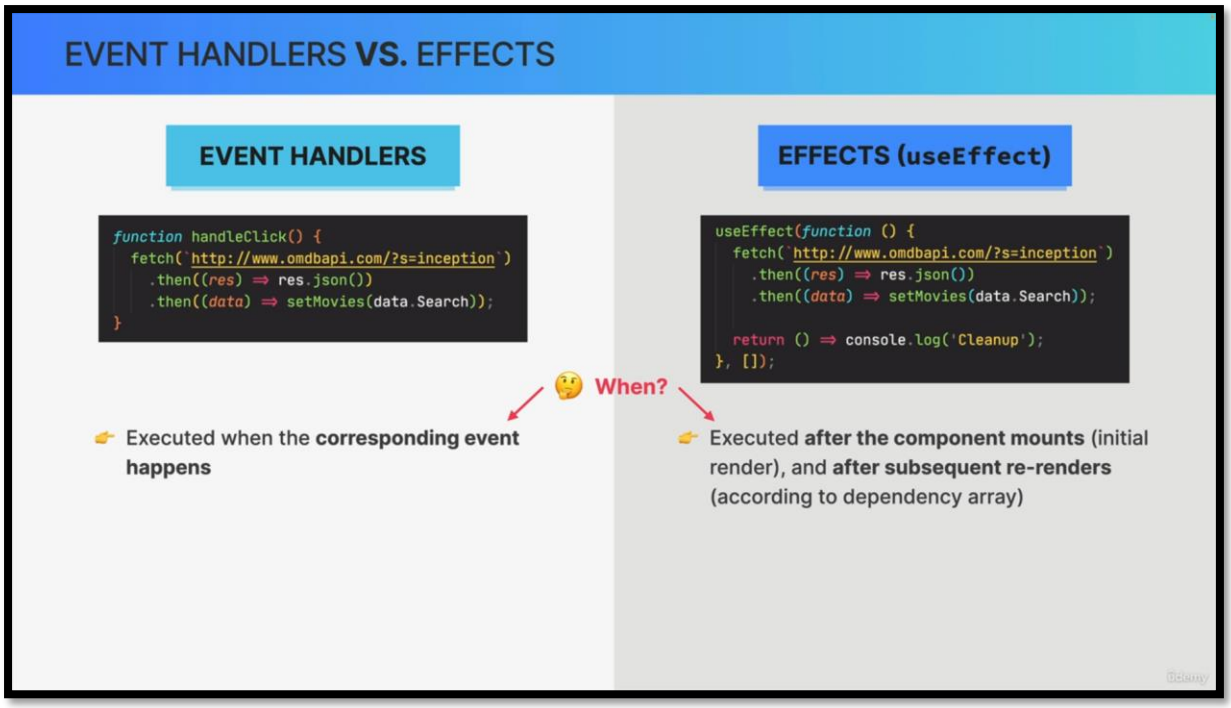
However, simply reacting to events is sometimes not enough for what an application needs.



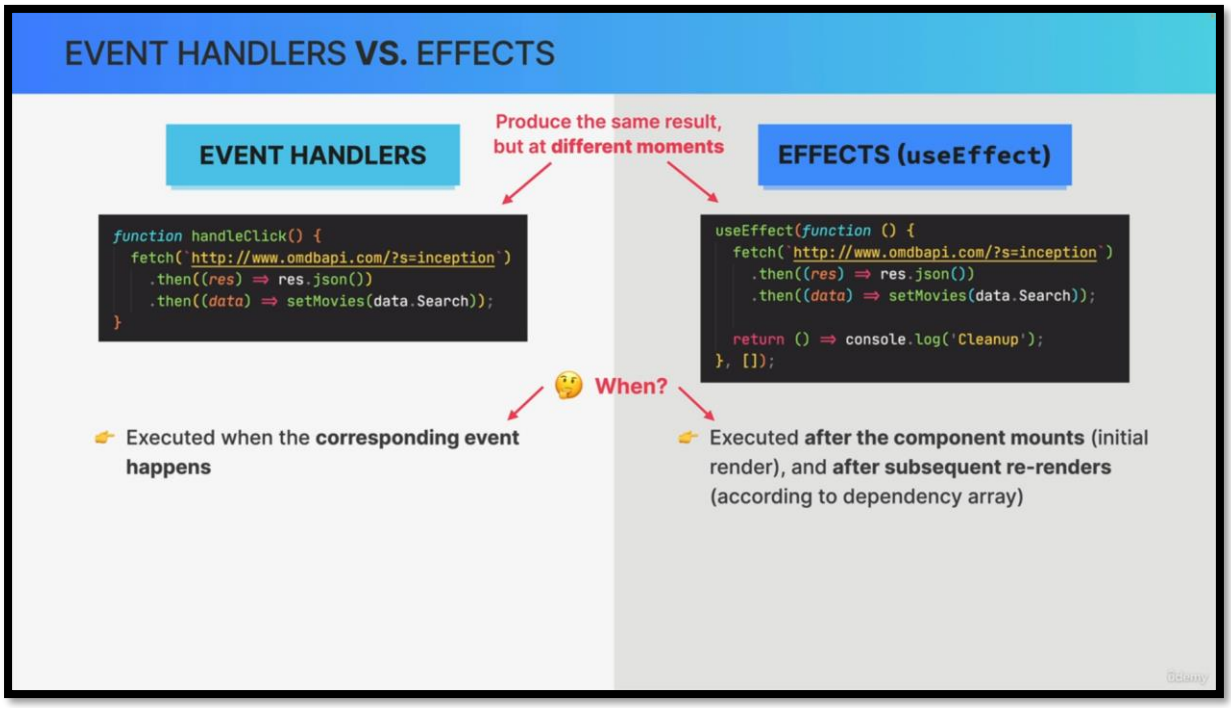
In some situations, we need to write some code that will be executed automatically as the component renders. Som this is when we can create a so-called effect by using the useEffect hook.



So, by creating an effect we can basically write code that will run at different phases of a component instance life cycle i.e. when the component mounts, when it re-renders, or even when it unmounts.

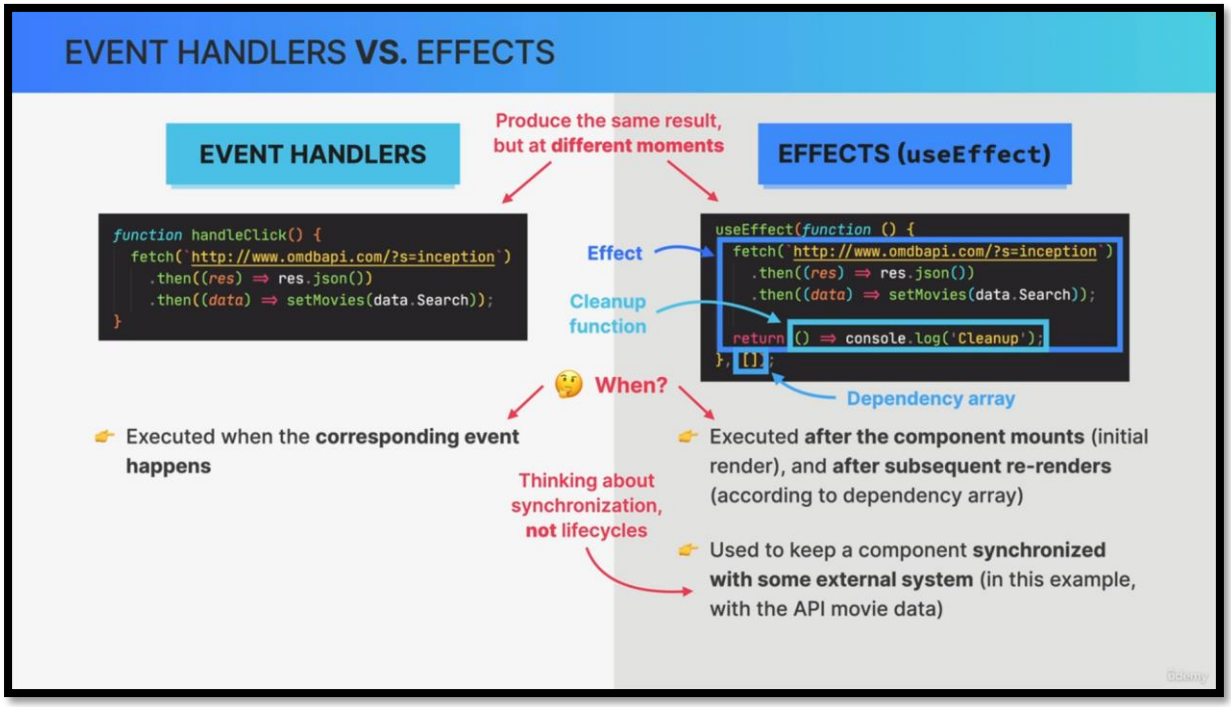


The first possibility is that we might want to fetch movie data only when a certain event happens. So, in that case, we will just use an event handler function. The other possibility of when to fetch the data would be to do so immediately after the component mounts. So, after it is first rendered.



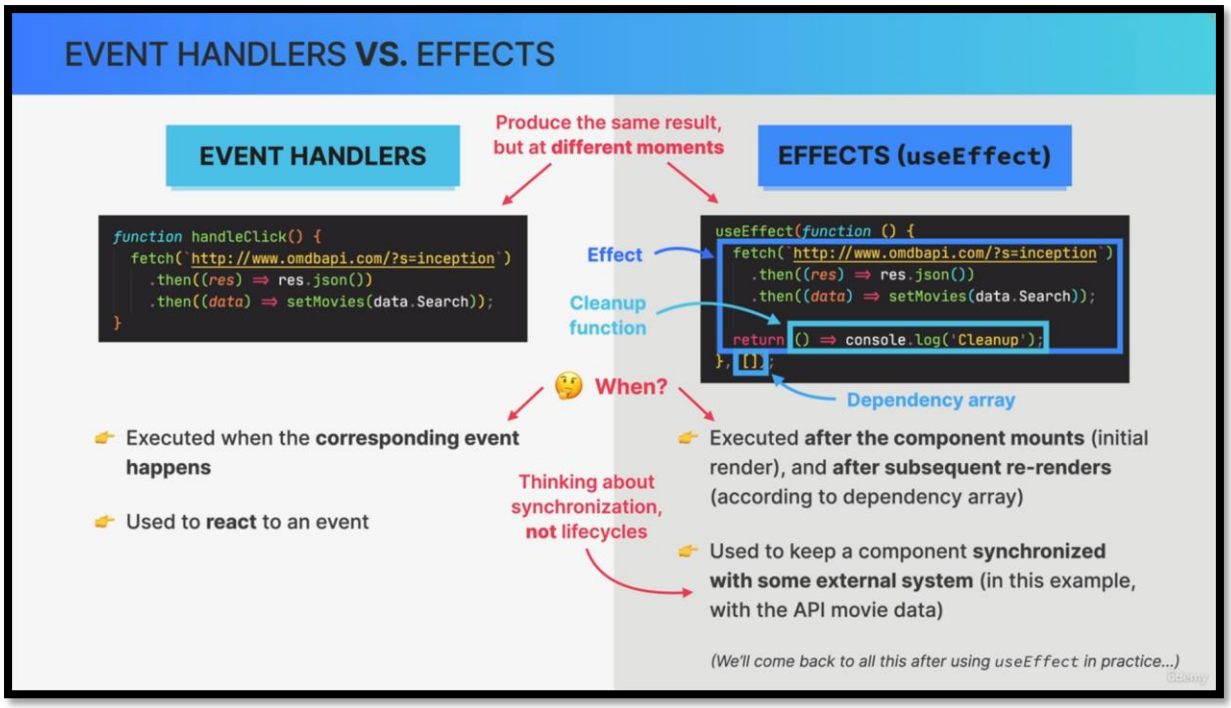
We can say that these two pieces of code produce the exact same result. So, they both fetch data about a movie but they do so at different moments in time.

The event handler executes when an event happens and the effect executes whenever the component first renders at least in this situation because the exact moment at which the effect is executed actually depends on its dependency array.



Now thinking about different moments of the component lifecycle ie mounting, re-rendering and unmounting, can be very helpful to understand how effects work. However, we should actually not think about life cycles, but about synchronization.

So, the real reason why effects exist is not to run code at different points of the life cycle, but to keep a component synchronized with some external system.



Event handlers are always the preferred way of creating side effects. So whenever possible we should not overuse the useEffect hook. So, basically everything that can be handled inside event handlers should be handled there.