

We are now ready to finally learn about how exactly React renders our applications behind the scenes. Now, there is so much to learn here that I split up this lecture into three parts. So it's this video and the next two ones. This one serves more as an overview, and then in the next two videos, we're going to go really deep into some React internals. And let's start with just a small recap so that we're all on the same page. So as we build our applications, what we're really doing is building a bunch of components. We then use these components inside other components as many times as we want, which will cause react to create one or more component instances of each component. So these are basically the actual physical components that live in our application and hold state and props. Now, as React calls each component instance, each JS will produce a bunch of react.createelement function calls, which in turn will produce a react element for each component instance. And so this react element will ultimately be transformed to Dom elements and displayed as a user interface on the screen. So we have a pretty good understanding of the initial part of this process. So all transforming components to react elements. However, what we don't understand yet is the second part of the process. So how these react elements actually end up in the Dom and displayed on the screen. Well, luckily for us, that is exactly what this series of lectures is all about. So in this lecture we're going to have a quick overview of each of the phases involved in displaying our components onto the screen. Then we're going to zoom into each of these phases to learn how the entire process works internally behind the scenes. So this process that we're about to study is started by react each time that a new render is triggered. Most of the time by updating state somewhere in the application. So state changes trigger renders. And so it makes sense that the next phase is the render phase. In this phase, React calls our component functions and figures out how it should update the Dom. So in order to reflect the latest state changes, however, it does actually not update the Dom in this phase. And so React definition of render is very different from what we usually think of as a render, which can be quite confusing. So again, in React rendering is not about updating the Dom or displaying elements on the screen. Rendering only happens internally inside of React and so it does not produce any visual changes. Now in all the previous sections, I have always used the term rendering with the meaning of displaying elements on the screen because that was just easy to understand and it just made sense, right? However, as we just learned, the rendering that used to mean is really this render plus the next phase. And speaking of that next phase, once React knows how to update the Dom, it does so in the commit phase. So in this phase, new elements might be placed in the Dom and already existing elements might get updated or deleted in order to correctly reflect the current state of the application. So it's really this commit phase that is responsible for what we traditionally call rendering, not the render phase. Okay. Then finally the browser will notice that the Dom has been updated and so it repaints the screen. Now this has nothing to do with React anymore, but it's still worth mentioning that it's this final step that actually produces the visual change that users see on their screens. All right. So let's now zoom into each of these different steps, starting with the triggering of a render. So there are only two ways in which a renderer can be triggered. The first one is the very first time the application runs, which is what we call the initial render. And the second one is a state update happening in one or more component instances somewhere in the application, which is what we call a rerender. And it's important to note that the render process really is triggered for the entire application, not just for one single component. Now, that doesn't mean that the entire Dom is updated because remember, in React rendering is only about calling the component functions and figuring out what needs to change in the Dom later. Now, again, this might seem confusing now because earlier in the course I made it seem as though react only rerenders the component where the state update happened. But that's because we were learning how react works in practice. And in fact, when we look at what happens in practice, it looks as if only the updated component is rerendered. But now we are learning how React actually works behind the scenes. And so now we know that React looks at the entire tree whenever a render happens. Finally, I just want to mention that a render is actually not triggered immediately after a state update happens. Instead, it's scheduled for when the JavaScript engine basically has some free time on its hands. But this difference is usually just a few milliseconds that we won't notice. There are also some situations like multiple setstate calls in the same function where renders will be batched as we will explore a bit later. So this is how renders are triggered, which is the easy part. What follows is the hard part, which is the actual rendering. And so let's learn all about that in the next video.