

Let's start this section with the conceptual difference between React components. Component instances and react elements. Knowing about this difference will hopefully make it a bit more clear what actually happens with your components as you use them. And also this is a pretty common interview question and so this topic is definitely worth learning about. And let's begin by taking another look at components. So components are what we write in order to describe a piece of the user interface. And a component is just a regular JavaScript function, but it's a function that returns react elements. So it returns an element tree, and we usually write these elements using the JSX syntax. Now a component is a generic description of the UI. So we can essentially think of a component as a blueprint or a template, and it's out of this one blueprint or template that react then creates one or multiple component instances. Now React does this each time that we use the component somewhere in our code. For example, the Tab component that we saw in the last slide is used. So it is included three times in this app component. And so therefore three instances of Tab are placed in a component tree. So in our actual application behind the scenes, this happens because React will call the tab function three times. So one time for each instance. So we can say that an instance is like the actual physical manifestation of a component living in our component tree. While the component itself is really just a function that we wrote before being called. And actually it's each instance that holds its own state and props and that also has its own lifecycle. So basically a component instance can be born. It can live for some time until it will eventually die. So it's a bit like a living organism, really. Now in practice, we many times just use the terms component and component instance interchangeably. For example, we just say component lifecycle and not component instance lifecycle. And we also say that a UI is made up of components, not of component instances, even though instances would technically be more accurate. Okay, so just keep that in mind in the future when you read documentation or some StackOverflow post or something like that. But anyway, as React executes the code, in each of these instances, each of them will return one or more react elements. So as we learned when we first talked about JS behind the scenes, JS will actually get converted to multiple react.createelement function calls. Then, as React calls, these create element functions. The result will be a react element. So a react element is basically the result of using a component in our code. It's simply a big immutable JavaScript object that react keeps in memory. And we will take a look at this later in our code. But what is this object actually? Well, a react element basically contains all the information that is necessary in order to create Dom elements for the current component instance. And so it's this react element that will eventually be converted to actual Dom elements and then paint it onto the screen by the browser. So based on all this, the Dom elements are the actual final and visual representation of the components instance in the browser. And again, it's not react elements that are rendered to the Dom. React elements just live inside the React app and have nothing to do with the Dom. They are simply converted to Dom elements when they are painted on the screen in this final step. Okay, so this is the journey from writing a single component to using it multiple times in our code as a blueprint, all the way until it's converted to a react element and then render it as HTML elements into the Dom. So I hope you found this interesting and useful. And if you did, then let's move on to the next video and take a look at all this in code.