As you work on your React application and on your React Components, you will notice in any project that your Components eventually become bigger and bigger, as you have more and more logic and JSX code in them. That's why React has this Component concept. This allows you to split your application into smaller building blocks, where every Component is focused on one core task. And then you build your overall user interface, by combining these building blocks.

By doing that, you keep every Component relatively small and manageable, and can build a complex user interface. There is no hard rule when to build a new Component versus adding more to an existing Component. But in the ExpenseItem Component we could already argue that it's getting a little bit big.

Nonetheless, we could argue that the calendar item which we want to show on the user interface could be treated as a separate Component. I personally like to split this ExpenseItem Component here into two Components, the ExpenseItem Component which we already have and then a new Component for rendering the date in the calender item look.

Therefore, we will create ExpenseDate.js. This Component will be a reusable Component and we can use it anywhere in the React application. This will now be shorter and therefore, maybe a bit easier to maintain and manage. And the bigger your application gets and your Components become, the more you want to look into splitting them up.

In ExpenseItem when we use ExpenseDate, we should set the date prop. Now the value which I pass here will just be {props.date}. And I know that this is now the point where it can get confusing, because now we're basically funneling some data through multiple levels of Components. In the end, we can see this Component tree come to life now, because now we don't just have the App Component and then one other custom Component, but we have multiple custom Components and they are nested inside of each other.

We could say, we pass data from App to ExpenseDate and we passed it on through ExpenseItem because that's how props work. We pass data from a Component to a direct child Component and we can't skip such a Component.