* -: This other way of handling errors could be interesting in cases like when you send a request and don't subscribe to it in your component.
* It's easy for us to react to an error in the component here, if we do return the observable and we subscribe in the component, because then we can pass this second argument to subscribe, which is this error-handling function.
* But what about creating a post? There I subscribe in the service.
* Now obviously you could just return the observable there too and simply subscribe in the component instead, and that would not be wrong.
* But you can of course also implement a different pattern of handling that error.
* You could use a subject, and that is especially useful if you have multiple places in the application that might be interested in your error.
* So for example here you could have your error property, which is a new subject and subject needs to be imported from rxjs.
* And that will, let's say eventually give us a string that error message.
* But of course you can store in there whatever you want.
* And the idea then simply is that here where we subscribe, where we get that error, we now use that subject.
* So this error we call 'next' and we pass on error message.
* And now the remaining step is that we subscribe to that subject in all the places we're interested in that error message.
* So here in this example, we would subscribe an ngOnInit in it for example, like this.
* And whenever we get our error message here, we could set this error equal to the error message.
* Now, of course, as you learned, it's a good and recommended practice to unsubscribe if you get rid of this component.
* So I'll add a new property error sub, which is a subscription and therefore we need to import subscription from rxjs, like this.
* And now we can store our subscription in that property here in ngOnInit, and now we just need to add OnDestroy here - this interface and import that from at @anglular/core as well.
* and add ngOnDestroy, therefore, maybe here at the end of the component.
* And there we now just want to unsubscribe, so this.errorSub.unsubscribe.
* Now we use a subject-based strategy for forwarding the error and that is fine too.
* information alert Schedule learning time Learning a little each day adds up.
* Research shows that students who make learning a habit are more likely to reach their goals.
* Set time aside to learn and get reminders using your learning scheduler.
* About this course Master Angular (formerly "Angular 2") and build awesome, reactive web apps with the successor of Angular.js