* -: No matter how you handle your error, what can be useful is a special operator that can assist you with handling errors.
* And that special operator needs to be imported from RxJS operators and it's named catchError, and it does exactly what the name suggests.
* Now let's say here, when we fetch posts, where we already piped some data, we got an error and we want to handle that.
* Now we can simply add the catchError operator here.
* We get our error response in here.
* So we get exactly the same data we would get in that second argument of the subscribe method.
* And in here, you could now do stuff like, "Send to analytics server," or anything like that.
* So some generic error handling task you might wanna do.
* Maybe not related to the UI, though of course, you could use the subject and next the error message here too.
* But maybe you have some behind the scenes stuff you wanna do when an error occurs, log it somewhere, send it to your own server, your analytics server, anything like that.
* And once you're done handling that error, you should pass it on though.
* It definitely needs to be able to reach Subscribe, just as you need to pass something here in Map as well.
* So, you also now need to create a new observable that wraps that error.
* And for that, you can import something from RxJS and that something is throwError.
* And that is a function that will yield a new observable and it yields a new observable by wrapping an error.
* So here we can now throw the error response.
* You could also throw a new custom error which you generated.
* Whatever you want.
* Important, you need to return that observable which is created by a throwError.
* And now with that, this of course doesn't do anything useful here, but it's just an idea that you could consider using catchError if you have some generic error handling task you also wanna execute.