* Now to fix that, we have two possible ways of doing that.
* The first alternative is that we use a subject in the post service where we next our posts when we got them and we subscribe to that subject in the app component and this would be a fine pattern, nothing wrong with that.
* The alternative and maybe a bit more suited here because the subject would be perfect if we have multiple components interested in the posts, that's not the case here, so a simpler and for this scenario, a better solution would be to simply return the result of our get method and of the pipe method thereafter and that would be our observable.
* So I don't want to subscribe here, instead I only return the prepared observable here in fetch posts and therefore right now no HTTP request gets sent because as you learned, requests are only sent when someone is interested.
* So here I'm not subscribing but now that I return this here, I can and I have to subscribe in the app component.
* So here where I call fetch posts, I can now subscribe again and I get my posts and now here we can set this is fetching to false and I want to set this is fetching to true here right before I start sending that request and even more importantly, we can set this loaded post equal to the posts we get back and this is now the code snippet I can also copy to onFetchPosts and yes you could of course also outsource this into a separate private method again to avoid this code duplication here if you want to.
* So now we moved the result handling so to say into the component but the more heavy lifting, the part detached from the template and from the UI which is the sending of the request and the transformation of the data, that now lives in the service and this is a best practice when working with Angular and HTTP requests.
* Move the part that is related to your template, which in my case here is managing the loading status and managing the loaded data, move that into the component and be informed about the result of your HTTP request by subscribing in the component but move the rest into the service and simply return the observable there so that you set up everything in the service but you can subscribe in the component.
* Now of course I'm using a different pattern for creating a post, there I am subscribing in the service and this can be fine too.
* If your component doesn't care about the response and about whether the request is done or not, as it is the case here in our application, if the component doesn't care about it, then there is no reason to subscribe in the component, then you can just subscribe in the service as we're doing it here but if it does care about the response and the response status as it does for fetching posts, then having that service component split is great.
* And now with that if this reloads, you see the loading indicator again, you see the posts again and now we have the best of both worlds and we have a setup which you will see a lot and which you will work with a lot when you're working with Angular.