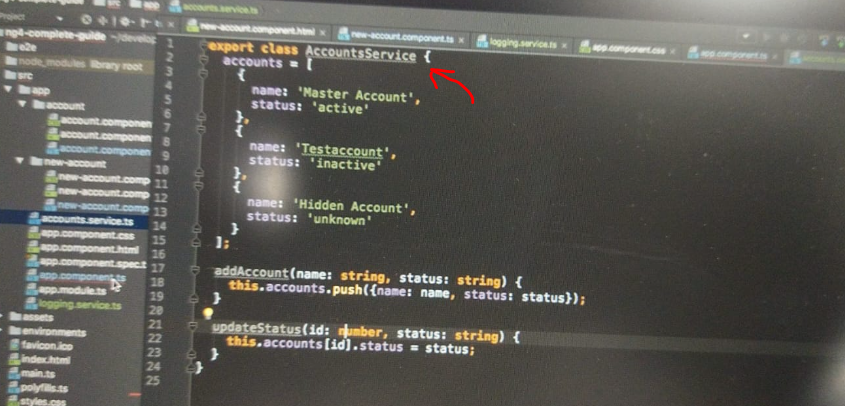
**108. Creating a Data Service**

* We saw how to use that loggingService in the last lecture, *now let me show you another typical use case for a service, to store and manage our data.*
* So for example our accounts, right now I store them here in the app component and then we have this chain of property and event binding to get data to the app component so that we there can update our accounts.
* Let's create a service for that, ***the account service*** maybe.
* So I will create this and I will export my accounts service class here, just like that and here I will now copy in my accounts, so I will remove them from my app component and paste them into my accounts service and I will add an addAccount method here where I simply expect to get an account name and status and I will add an updateStatus method where I expect to get the ID of the account I want to update and then the new status here.
* Now the logic for these two methods is basically the same as in the app component, for a new account or in the add account, I simply want to push a new account onto my array of accounts and here, I will simply quickly create a new object where I assign the name and the status, so the data we get passed into this method and for the status changed, well here I basically want to also just access my account with that ID here and set the status to the new status we get passed as an argument too.



* So with that, this service looks all right, we got our accounts there and we got our methods to change something there, of course the app is broken though because for one, I don't need that code in the app component anymore.
* If we have a look at the app component template, we still loop through all the accounts here, so I should still have my accounts here in the app component but we remove them from here, they're now in the service.
* So we should maybe re-add them here but now, this should simply be an array of accounts, so each account object has a name which is a string and a status which is a string, so I'm just defining the type here and we want to have an array of such objects, initially it should be an empty array, this is what I'm saying here.
* Now since we have our account service, we should inject it here as we learned it.
* So let's add our accounts service here, of type accounts service, the type is super important to inform Angular about what we need here.
* Let's close this body of the constructor and make sure to add the import at the top pointing to your account service file.
* Now what did we learn? We also need to add a provider.
* So here at the providers array, we should add accounts service so that Angular knows how to create such an account service.
* With that, we can now add for example onInit, this onInit lifecycle hook as most initialization should not be done in the constructor but instead here and simply set this accounts, accounts equal to this account service which is available as a property because I use this shortcut with the accessor in front of the argument name and here, I can access my accounts field.
* Just as a little note, since accounts here is an array, it is a reference type, this is normal Javascript behavior.

Text

Description automatically generated

* So this is a reference type, so by setting it equal here, we're actually getting access to the exact same array as stored in the service, just as a side note.

Text

Description automatically generated

* So with that, we do have access to the accounts and if we save this and have a look at our application, this still seems to work.
* These buttons will not work and will actually give us errors because I remove the event listeners but in general, this looks all right, we can see the three accounts which are now stored in our accounts service.
* So with that, let's go back to our application and here, I now want to update the account component and the new account component, let's start with the new account.
* Here we no longer need to emit this event here because we're not listening to it anyways.
* So we can get rid of the output here, get rid of the imports up here, of eventEmitter and output, instead we should inject our service, our accounts service of type accounts service.
* And what did we learn? We should provide it, right, so let's add account service, account service and make sure to add the import at the top.
* Now with this, in here I can call accounts service, add account and pass the account name and the account status as this is what this method expects to get as arguments.
* So with that, let's go to our account component now, let's also get rid of the output here.
* We still need the inputs because we're still receiving that data from outside and we can get rid of this event calling here, we removed the output anyways, get rid of the imports we don't need anymore and now here, I also want to get access to the accounts service.

Text

Description automatically generated

* So let's inject it here too, maybe in a new line to make it more readable, the accounts service of type accounts service, like this, make sure to add the import at the top and let's add it to the providers array here.
* With that, I can now call it here, account service, call update status, pass my ID and the new status, again this is what this method expects.
* If we now save this and go back to our application, it looks all right but we quickly realized that if we click on add account, we don't get an error, we get the log but I don't see the account.
* If I click these buttons, I also see the log but I don't see the change here, so something is not working correctly, it seems like it doesn't, right.
* Well let's have a look at what's going wrong here in the next lecture.