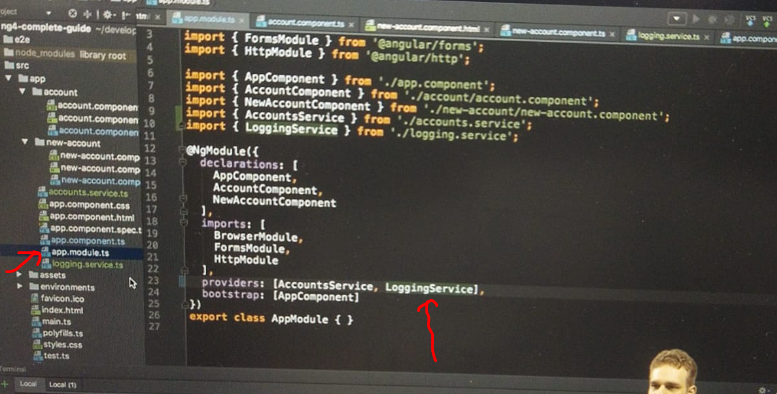
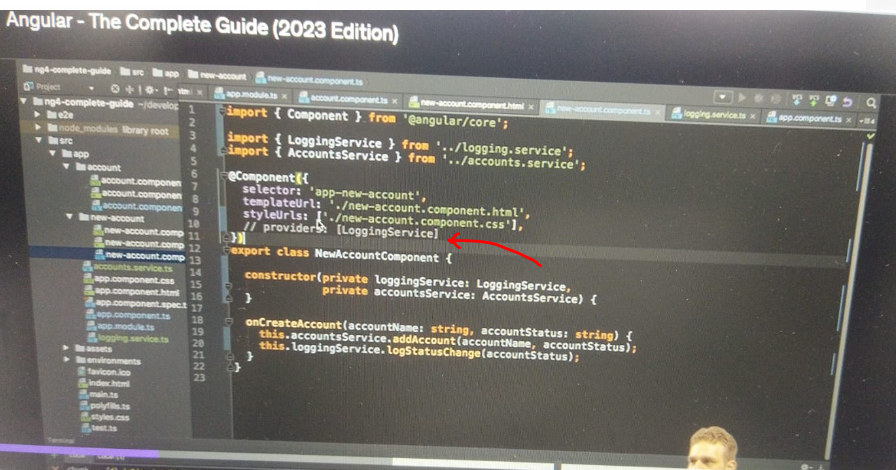
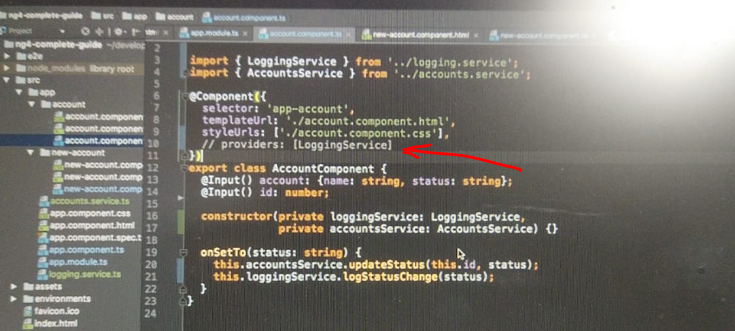
**111. Injecting Services into Services**

* -: In the last lecture, we learned about this hierarchical injector, and that we can choose between one instance or multiple instances of the same service class.
* I mentioned that the highest possible level is actually not the AppComponent.
* So let's remove it here, this providers array, leave it in the the constructor and leave the import, but the highest possible level is in the AppModule.
* Here, we also have a providers array added if you don't have it, and here we can also provide the AccountsService.
* Make sure to also add the import at the top here, then.
* *With that little addition here, we're making sure that our whole application, everything in our application receives the same instance of the service unless it overrides it as we did before.*
* **Injecting Services into Services:**
* *Now with that, we can now even inject a service into a number of service, because that's not possible* ***by providing it on a component level, for that we need to provide it in the AppModule****.*
* So, let's say we want to log something, if we call action in the AccountsService.
* So I would have to provide a LoggingService in my AppModule now, and therefore I will simply comment out providers here in my @Component decorators.



* And I don't want to call LoggingService here anymore, so we'll comment this out and I will actually leave the injection here so that you have it for reference if you download the code.
* I'll also comment it out here in new account component.



* 
* Instead, let's say I want to use this LoggingService in the AccountsService, and I might want to call the log status change method there because this isn't the enter place where the status changes, either because we added a new account or because we called update status.
* ***So to use the service there, generally we start in the same way.***
* *We add a constructor and we inject our LoggingService.*
* In this case, set the type, import this type at the top, make sure to add the import.
* And we provided it an AppModule.
* So you could think that we are set to go, that we can now go to add account, call LoggingService, log the status change, pass the status here, and do the same here for updates status.
* You could think this is how we do it and it's not unlogical to think so, but let's see if this works.
* If we load our app, we already get an error.
* "Can't resolve all parameters for AccountsService.
* " Something's not working.
* ***The reason is, if you inject a service into something, this something needs to have some metadata attached to it.***
* Now, a component has some metadata because we have @Component.
* A directive has some metadata because we have @Directive.
* Account or, excuse me, a service doesn't have any metadata.
* It's just how Angular works that we need to attach some metadata, and of course, not any metadata.
* You shouldn't attach @Component here.
* There is a specific metadata you should attach in such cases.
* **@Injectable:**
* ***It's @Injectable, which needs to be imported from @angularcore.***
* This tells Angular that now this service is is injectable or that something can be injected in there, to be precise.

Text

Description automatically generated

* So that's important.
* *You don't add @Injectable to the service you want to inject, but to the service where you want to inject something.*
* *So the receiving service you could say, that to be injected in service.*
* And you don't need to add it to any other service.
* If you don't want to inject anything in a service, like here for the LoggingService, you don't need to add @Injectable.
* Only add it if you expect to get something injected.
* **Now, one important note here**, as I explained, we don't need to add injectable, the injectable decorator here to this class, because this is a service which we only inject somewhere else, but where we don't inject anything into this service.
* And you technically really only need injectable if you inject something into this service, not if you just inject this service somewhere else.
* ***In newer versions of Angler, however, it is recommended that you always add @Injectable.***
* It technically makes no difference right now, in the future it might though, so it's already a habit you might get into that you add it.
* Here, I do omit it, but you could consider adding it also, to ensure that in the future, it doesn't unexpectedly break.
* What I explained about why you need it and that you don't need it here technically, still holds though.
* With this, if I now save this and we let the app reload, now it works, and now you see that if I save this, we still log it to the console with the correct status.
* The same happens if I click here, but now everything is centralized into our service, and now we're also injecting a service into a service.