**111. Injecting Services into Services:**

* -: In the last lecture, we learned about that hierarchical injector.
* And actually, this injection behavior leads to the behavior we see here.
* Now, that's not a bug, that's intended.
* You might have an application where you want to have many different instances of the same service, where you absolutely don't want to have the same instance.
* But sometimes you want to have the same instance, and we definitely do.
* Because right now for our AccountsService, we have three instances.
* The first one gets created here in AppComponent, we provide it here.
* AppComponent receives its own service or its own instance of the service, I should say, as to all the child components.
* Now you might say, "Fine," because new-account and account are child components, right? But there we provide it again and hence we override the instance we would get from the AppComponent.
* Now, new-account and account.
* component have their own instances.
* So these are the other two instances of this service class.
* And therefore if we add a new service, we call addAccount.
* There we push it on this array of accounts and this actually works but it's a totally different instance of the service than the one we use here in AppComponent, the one through which we loop.
* ***How can we fix it then here for our app?***
* *Simple, we just remove it from the providers array.*
* *Don't remove it from the constructor.*
* *We of course do need it here because that tells Angular that we want an instance, the providers array basically tells which instance so there we needed to remove it.*
* *And the same for account.*
* *component, leave it in the constructor, leave the import at the top, but remove it from the providers array.*
* With that little change, if you go back to the application after saving it, let's add another account.
* Now you see it at the bottom, now you can change the status again.
* So this is the difference, different instances of the service.
* Now we're using one of the same before we weren't.
* Both may be the behavior you want in your app but make sure that you get the correct one.