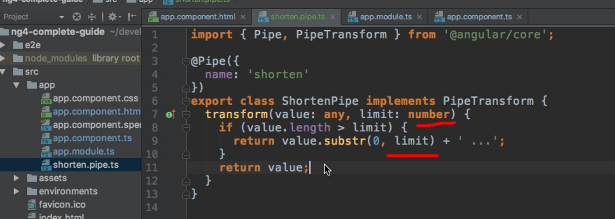
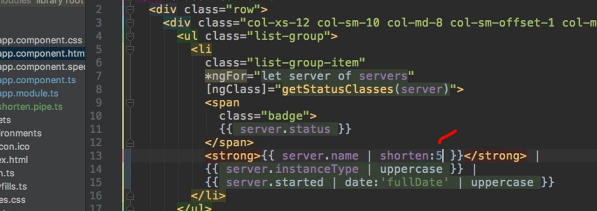
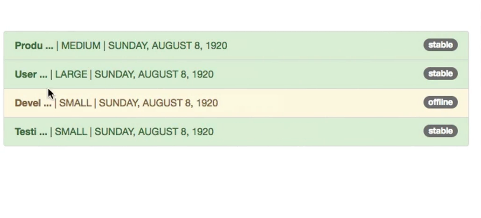
* -: In the last lecture, we created our first pipe, the shorten pipe, and it's already working quite well.
* But it would be nice if we could allow the user to specify the number of characters, at which point you want to shorten the pipe or excuse me, shorten the value.
* Right now we're always using 10 characters as a limit.
* This is hard coded into the pipe and this might not be the best pipes you can build.
* Instead, it would be nice if we would receive a second argument into the transform method.
* D, let's say limit.
* And this should simply be a number.
* Now limit would be what you then use here in the check.
* So only if it's greater or if it has more characters than this limit, you want to shorten it.



* And then of course, you use limit here in your substring method too.
* Now with this little change, you allow the user to pass a parameter to the pipe because now you receive a second argument in the transform method.
* And since you receive this second argument here the user may pass one parameter to the pipe.
* You don't need to change anything else.
* Don't need to change anything in the decorator.
* This is all.
* Now, with this, we can go to our app component and if we would leave it like this, you see it's not short at all because we're not passing any limit, therefore it's undefined.
* Therefore, we always fail this check whether it is too long.
* Therefore, we always return the original value.
* But if we do pass a parameter, and remember we do this ***by adding a colon and then the value***, let's say five if we do this and save this.
* Now you see we're shortening after five characters.
* 
* And if you pick 15, now you see the first two names go through quite well because they are short enough.
* But these last two names are shortened.



* And this is how you can of course, add multiple parameters.
* You would add another parameter simply by adding another argument here.
* So this would be anotherArg of the type you want it to be, whatever it is.
* And then with that added, you could simply add a number colon and your second parameter.
* This is how you parameterize your own pipes.
* And of course, you could also chain them with built-in types and so on.
* You can use it like a built-in pipe.
* It's just your own pipe.
* So this is awesome.
* Let's now move on to an advanced use case and to some limitations you might face when using pipes.