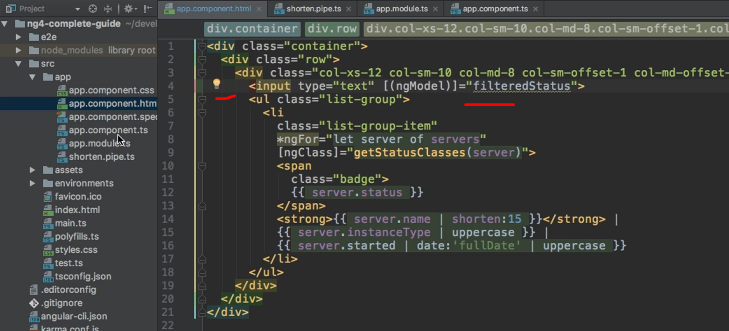
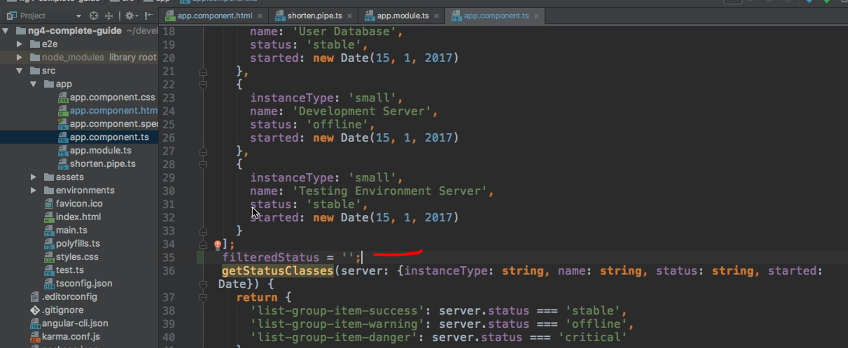
* -: In the last lecture, we created our own pipe and we also allowed the user to pass a parameter.
* Now I want to create another new pipe because I want to show you something which might look strange at the beginning.
* Let's add a new element here above our unordered list, and this should be a input field.
* So input of type text.
* ***I want to allow the user to filter our servers***, for example by status so that once you type stable in there, you'll only see servers with the status stable.
* So therefore, I need this input text field and I want to use two-way data binding to bind it.
* So ngModel and bind this to filteredStatus for example.



* This of course would be a new property I set up here in my component.
* So below all my servers I have filteredStatus, which is an empty string at the beginning, filteredStatus like this.



* So if we save this, we should see this new input field here at the top.
* Let's maybe add a horizontal line below it just for styling reasons.
* And with that, we can enter something here but of course this does not do anything.
* *Now I want to build a pipe which I can apply to this list, somehow, which allows me to then only view the servers which do indeed fulfill the requirements here.*
* So to do this, I will add a new pipe here.
* And for this I will now use the CLI.

Text

Description automatically generated

* So ng generate pipe would be shortcut here or just g p.



* And then the name of your pipe, and I will name it filter, filter pipe.
* So by typing filter this gives me all the testing file, which I don't need but then the filter pipe here.

Text

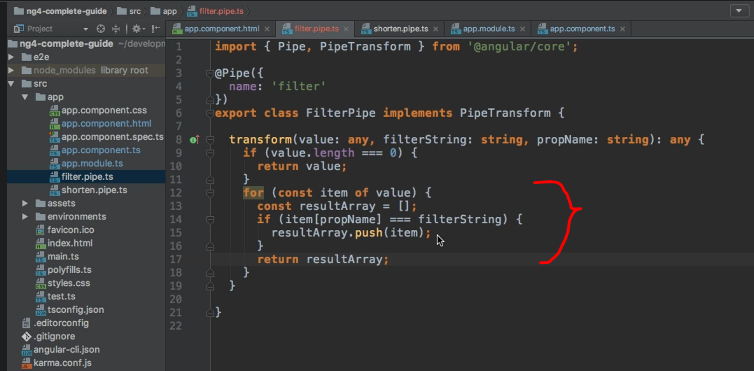
Description automatically generated

* And there you can see it also pre-populates the name here in my @Pipe decorator.

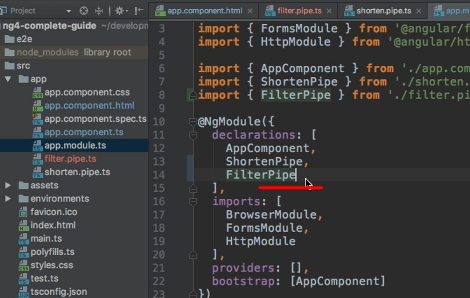
Text

Description automatically generated

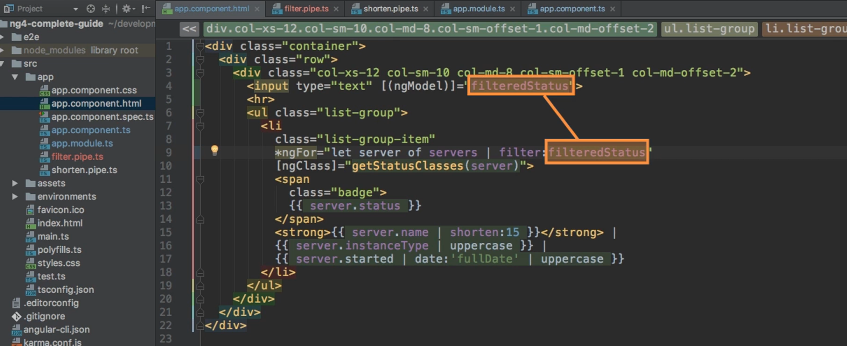
* And it gives me this interface and, well, it gives me this transform method in its most basic form.
* So in here, I want to allow the user to filter, and I will indeed get an argument.
* The first argument of this list of arguments and this will be what the user entered.
* So the filterString, you could say, and this will be a string.
* Well, and then here I want to implement some logic which allows me to only return the elements of the array which fulfill this argument or which fulfill this filterString here where the status of the server matches this filter string to be precise.
* For this, I will first of all check if value.
* length and value will here be the array of service.
* It doesn't have to be a string, this is the first important takeaway.
* It can also be an array here, it can be any data you output in the end.
* So here I want to check if the value length is zero, so if it's empty.
* In which case I will just return value, so I will basically just quit, there is nothing to display.
* And now after checking this, the interesting part happens.
* Here, I simply want to check or I want to loop through all my items in my value, and value here will be an array, keep this in mind.
* And then I want to check if my status of each server matches the filterString up here.
* So for this, I can simply check if item.
* status, and of course you could write this more generic and pass the to be filtered property here as a second argument like propName, which is a string.
* And then you could simply say, item propName in square brackets.
* So if this equals my filterString, only if this is equal, only in this case I want to return it.
* So I should add a new result array, this is an array I will use temporarily here, and I will push new items on this result array.
* So I will push any items on this array which fulfill my criteria here, which are or which have a status or a prop name in general equal to my filterString.
* So here I will then push item on this array and return this result array in the end.



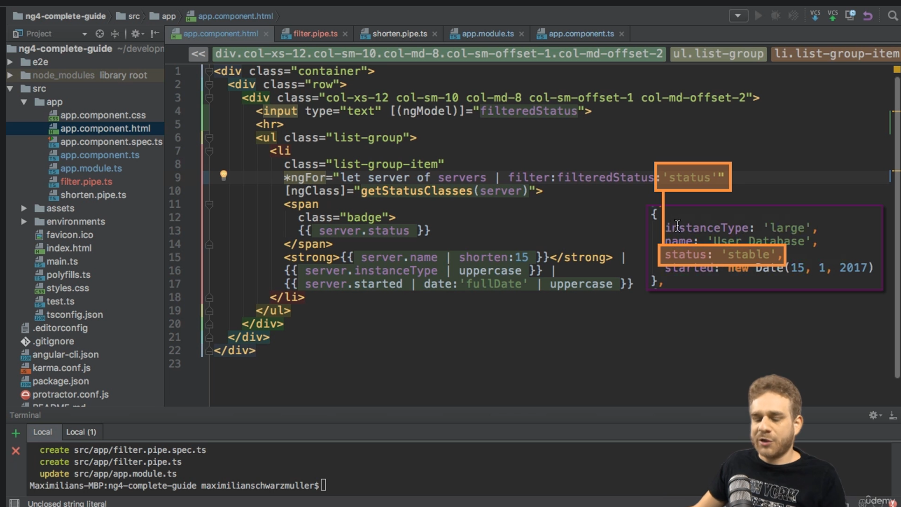
* And therefore this result array will only hold the items where the propName I passed as a second argument here, or as a third argument actually, where this propName or this value of the propName is actually equal to my filteredString.
* Now with this, the pipe is finished.



* You have to make sure that it has been added to AppModule, I did this automatically here for me.
* If you created it manually, make sure that you add FilterPipe to declarations and add your import at the top.
* **And then in the app component, we apply it here in the ng for loop.**
* ***And this might sound strange because: before, we only used it in string interpolation, but keep in mind that I said at the beginning of this module that pipes transform your output and the ng for loop is simply part of your output.***
* Therefore, of course you can add a pipe here to my servers.
* And then I simply say filter, of course we need to pass two parameters here.
* The first parameter is the filteredStatus, this property which holds my string for which I want to filter.



* And the second argument is my propName, so this will be a string, and this string should be status because I want to filter on the status property.



* Now if we save this and reload the app, you see no servers are displayed by default because no servers match this criteria.

Graphical user interface, application

Description automatically generated

* If I type stable here, you see we see the stable server, but that doesn't look correct because we should have more than one stable server.
* If we have a look at our app component, we get free stable servers.
* So we only view the first one, and the reason for this is simply that right now here I'm returning inside of this for loop, so this is a little logical mistake here.

Text

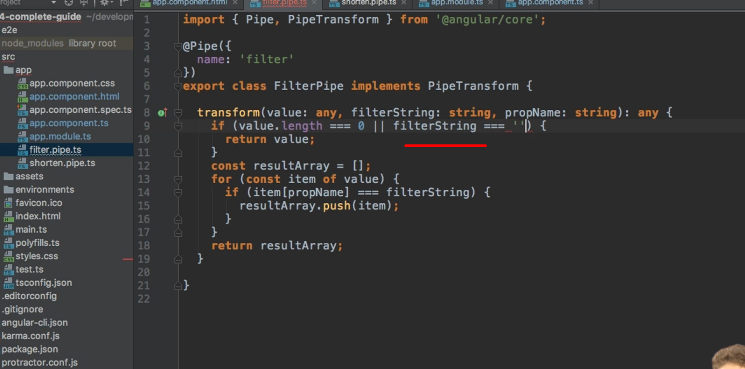
Description automatically generated

* We should of course return outside of it, and also create this array outside of it like this.
* So with this change in place, now we should see that once we add stable, we see all stable servers.
* And once we add offline, we see the offline server.
* So our filter pipe is now working as expected.

Graphical user interface, application

Description automatically generated

* Now, if we wanted to see all servers, we need to adjust this code that in case our filterString is empty.



* So we could add this here as a OR condition at the top if filterString equals an empty string but then we see all servers, and now we are able to filter for whichever servers we want to see.

Graphical user interface, application

Description automatically generated

* This works great but there is one issue, we will dive into in the next video.