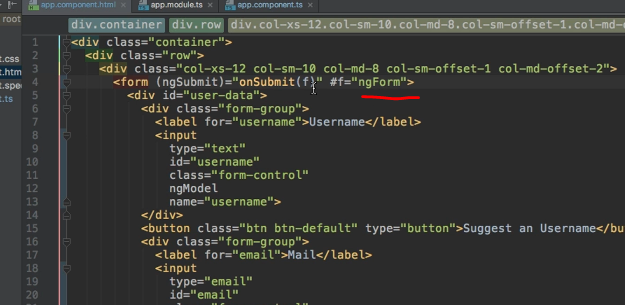
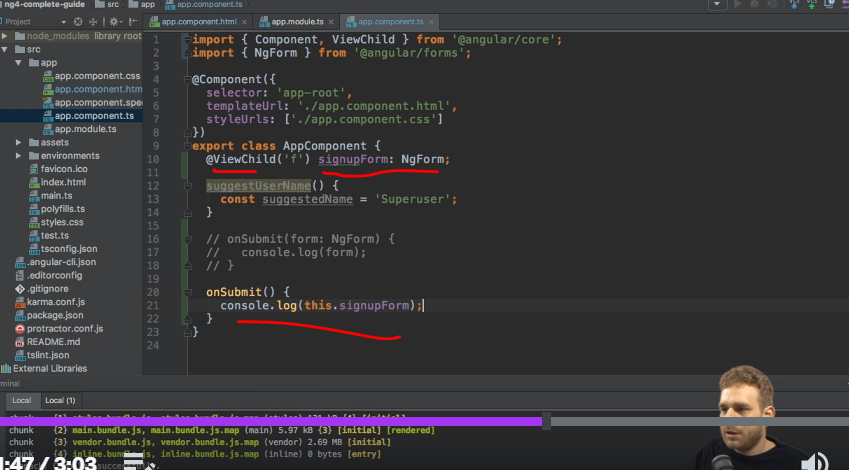
**189. Accessing the Form with @ViewChild:**

* Instructor: In the last lectures we learned how we can register controls and how we can submit the form and also which properties this form has.
* Now, right now we submit the form by passing the form which we get via ngform here to the onSubmit method.



* This is absolutely fine and probably the approach you will use in many use cases.
* I just want to bring some other approach to your attention.
* You don't have to submit it here.
* Remember the component section where we learned about at ViewChild, which allowed us to access a local reference, an element controlled or which holds a local reference in our TypeScript code.
* Well, in the end we do just have such a local reference here and whilst it might not point to an element ref but to the ngform object it still is a local reference in our template.
* So we can also use at ViewChild here.
* So I will simply comment out this onSubmit method here so that we have it in the code you can find attached to this module and show you an alternative approach.



* In this alternative approach I will use @ViewChild, this decorator you learned about in the component section.
* So make sure to import it from @AngularCore and I want to get access to the element which has the local reference f on it.
* So I pass f as a string as an argument to ViewChild and then I could simply store this in a variable named signupform, any name you like, which will be of Type ngForm, of course.
* Now in onSubmit, I could output this signup form like this and you should see that if I now submit this again we still have this form, and I can also enter something so that we can see that this works too if we have a look at the value.

Graphical user interface, text, application

Description automatically generated

* So this gives us access to the very same form without passing it to onSubmit.
* This is especially useful if you need to access the form not just at the point of time when you submit it, but also earlier and I will show a use case for this in a later lecture.
* For now, let's keep in mind that this is another way of getting access to the form in our TypeScript code, a perfectly valid way of getting access.
* And before diving deeper into why this might be useful or when we could use that, let's actually understand how we can control the validity of the form, so to determine whether the form is valid if we enter the valid email address here or not.
* Because right now I can't submit anything, no matter if this is invalid, if this is empty, it would be nice if you could add such validation and take advantage of the tools Angular gives us there to possibly also enhance the user experience by placing a red border around invalid elements or something like this.
* We'll have a look at validation in the next lectures