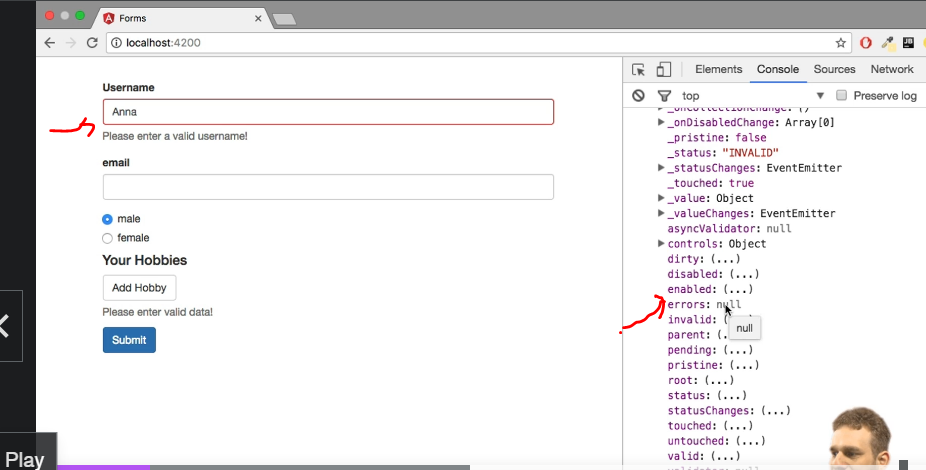
* -: In the last lecture, we added our own validator and we saw this strange error code.
* Let's see what it is up with this error code.
* So right now this form is invalid.
* This field is invalid and the overall form is invalid.
* I can still submit it though because I haven't disabled the button.
* So we have a look at this, we see the errors property on our form.
* If we open this, it's null, though.

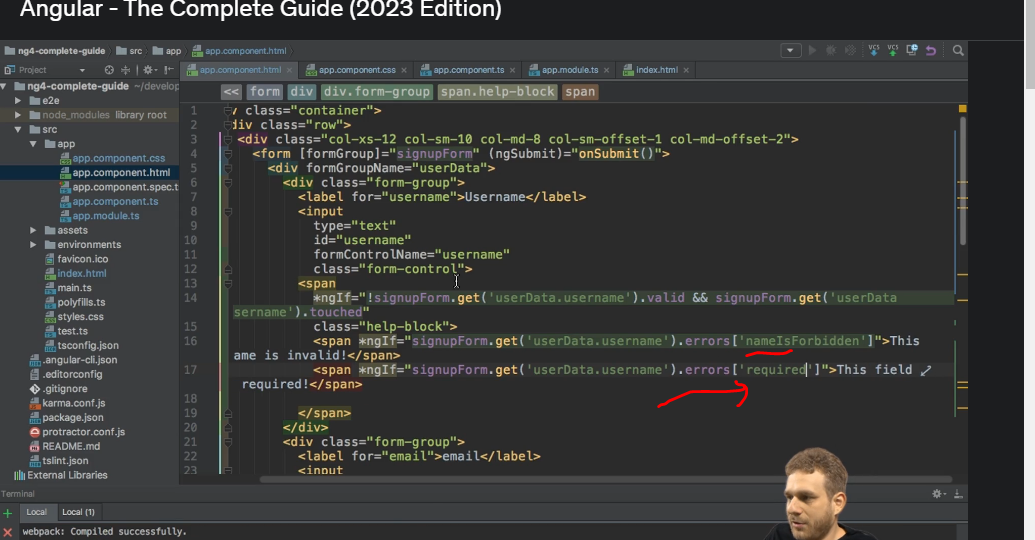


* The interesting part is if we have a look at our controls here, for example, user data and then there on the controls, the user name, here if we have a look at the errors, you see the name is forbidden True key value pair.

Graphical user interface, text, application, email

Description automatically generated

* So this is where Angular actually adds the error codes on the individual controls on the errors object.
* So we could take advantage of this by going back to the HTML code and fine tuning this error message.
* Let's say we want to say the username is required if the field is empty and we want to say this is an invalid username if it is invalid.
* So we could say this field is required here, our current check will not suffice though because here we only check if it's valid in general.
* So here we could simply wrap a number span in there and check if sign up form, get user data user name errors.
* If this has the name is for forbidden error.
* If it does have this, I want to say this name is invalid.
* I'll also add another span.
* And here we need to find out what the actual message is if the field is required.
* So I'll enter the text here for now, but then we need to check what the error message would be.
* So let's quickly submit this form here and check on our controls user data and there on the controls, the user name.
* And there again, the errors.



* We see here the code would just be required so we can now check if it contains the required error.
* With that in place, if I now click in here and out of there, we see the field is required, if I type Anna in there, we see the name as invalid, and if we type something valid, we don't see anything.
* So this is how we can use these error codes.
* And of course you could all use NG switch here or any average setup.
* The key thing is to understand that these error codes can be used to show the right error messages.
* And of course you can come up with even more complex setups where you have TypeScript objects in your TypeScript code, where you map error codes to specific messages and dynamically output that here.
* This is the basic example, the basic way of using these codes.