* -: So in the last lectures we added validation.
* Now as in a template driven approach we can now use this **form status** to display messages but it works a bit differently because we access the controls differently.
* Let's say we want to display a message below the input here for the username, if that is incorrect.
* So I'll add a span with a class of help block, again, that bootstrap class, and it'll say please enter a valid username like this.
* Now, in a template driven approach, we would place username on this and say model to get the reference.
* This doesn't work here because again this is not set up why our NG model.
* This is not registered by our NG model but it's still very easy.
* We can simply add NGF of course because we want to determine whether to display this or not.
* And then we can get access to this by accessing our overall form.
* And here we have a GET method.
* The GET method allows us to get access to our controls easily.
* Here you can either specify the control name or I will come back to this later, the path to the control.
* Now for now the path is the name because we only have one level of nesting in our form object, and here the name is username.
* So this gives us access to this username control.
* And now valid holds whether this control is valid or not.
* So if it is not valid I want to output this and of course, again to give the user the chance to change it.
* I will all check if signup form get username actually was touched.

Text

Description automatically generated

* So by adding this, we should see no messages displayed.
* If I click in there and click out of there, we see the error and if I enter something valid, it's gone again.

Graphical user interface, text, application, email

Description automatically generated

* So this is how easy you can get access to a control in your form with this GET helper method.
* Now I will repeat this for the email here.
* So below the input here, I will simply exchange username for email and I will do do it for the overall form.
* Here I can simply remove this, get access here, and just execute valid and touched on the overall form, which of course also has these properties.

Text

Description automatically generated

* So please enter valid data might be the right message here.
* And up here for the email it should be please enter a valid email.
* So with this in place, we should see nothing right now.
* If I click in here and out of there, we see the message for the username as before and for the overall form which is still there because email is invalid.
* But now if I enter something valid there, we don't see it.
* If we remove this, we see the message for the field here and for the overall form again.
* So now we're using the form state again.
* And of course, keep in mind these CSS classes NG touched, NG invalid are still added.
* So you can still go to your CSS code and say A input with NG invalid and NG touched should still receive a border which is red, for example.

Graphical user interface, text, application

Description automatically generated

* So that is still possible.
* The classes are the same as in the template driven approach.
* So now if I click in there and leave it you see it now has a red border which disappears as soon as the value becomes valid.
* So this is how you can still use the form state.
* Important takeaway, you access your elements differently with this GET helper method to conveniently get access to them.