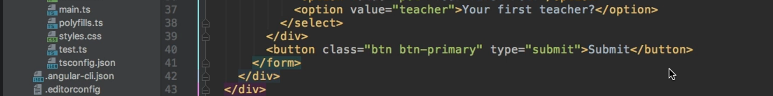
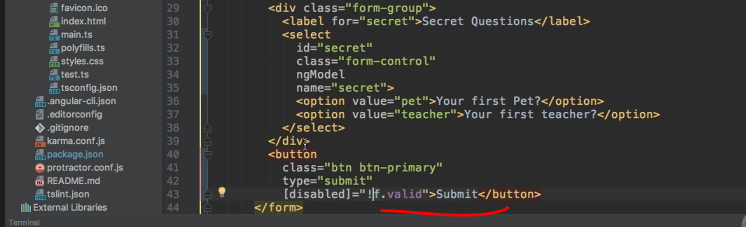
**192. Using the Form State:**

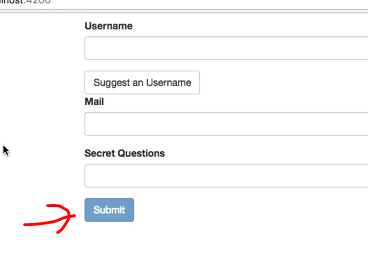
* -: In the last lecture, we found out that *Angular tracks the state of each control of the form, whether it's valid* and so on.
* And conditionally also adds these CSS classes.
* Now with that information, we can go back to our form and you take advantage of it.
* Before diving into the CSS classes.
* The easiest way of taking advantage is down here on the submit button.



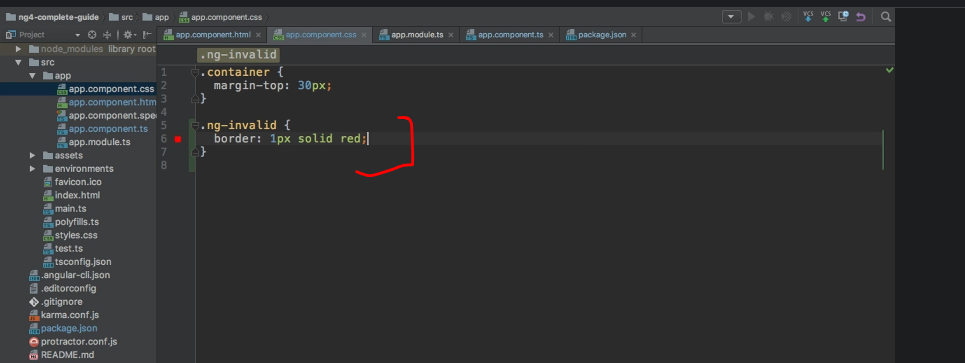
* *Let's disable the button if the form is not valid.*
* So I'll just split this over multiple lines and add some property binding.
* *I want to bind to the disabled property which will add the, or which will set the disabled state of this button to true or false, depending on some condition.*
* And I will specify the condition here.
* Now, I could say true here to always disable it.
* That of course is not super useful.
* Instead here I want to check whether the form, remember we do have access to it on this F local reference here.



* So if this form is valid, would you be precise if it is not valid? So if this form is not valid, the button should be disabled and we can actually see this, once this reloads that now the button is disabled, and only if I enter some valid values here I can hit the submit button again.



* So this is the first improvement in place.
* ***The second improvement is to take advantage of these CSS classes.***
* We do get these classes added automatically.
* So we can now go to the style sheet of this app component and we could say that on NG invalid we want to give it a red border.

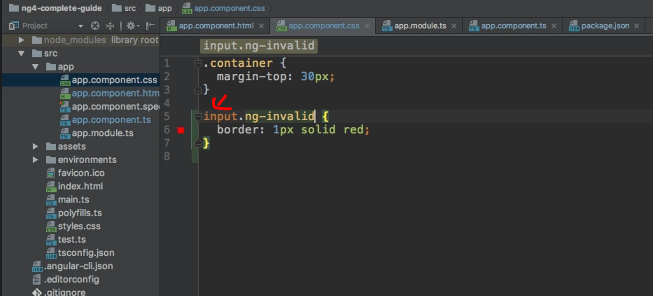


* So one pixel solid red.
* Sounds like a solid idea, doesn't it? Well, if we do this, you see everything's red now.

Graphical user interface, text, application, email

Description automatically generated

* The two controls, but also the overall form.
* ***The reason for this is, that our overall form is also invalid and that Angular also adds the NG invalid class CSS class to the form element.***
* *So a better approach would be to make sure that it is not added to the form.*
* Now, there are a couple of ways of doing this and it's all just pure CSS logic.
* One idea would be that we want to be inside of the form element, but then it will still place these red borders around grouping devs.
* We will have a look at what I mean with this later.
* ***So the best way is, to simply be explicit that we want to add it on inputs,*** for example.
* And of course you could also add select here.
* So input with NG invalid or select with NG invalid, and which other elements you have.
* Just be creative here.
* There are different ways of achieving the skill.
* In the end, you just want to make sure that the border is applied to the right controls.

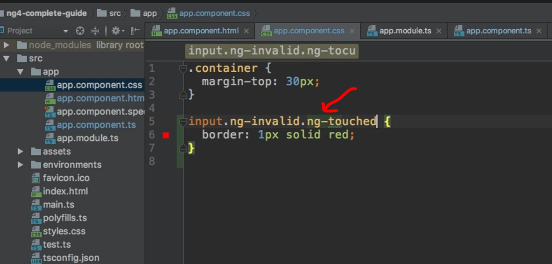


* Now, with this in place, we see that now we only have a red border around the invalid controls, ***but we do have the border right from the start,*** which is also not great because, I at least want to give the user the chance of changing it before showing a warning showing that it's wrong.

Graphical user interface, text, application, email

Description automatically generated

* So a ***better approach might be to make sure that we only add a red border to input which has the CSS class NG invalid and also the CSS class NG touched.***



* So that user has to at least click into it.
* Now, by default, we don't see anything red even though the form is invalid.
* But if you click in there and decide, Yeah, I'm done.

Graphical user interface, text, application, email

Description automatically generated

* This is my value.
* Well, now you see we have a red border because now we had a chance of changing it.
* We didn't change it.
* It is invalid and we want to show this.
* And with that, we're taking advantage of this form state handled by Angular.
* We disable the button, and we show a visual feedback to the user.
* Now, you could of course go much further.
* You could also add a warning message below this input here, for example, and output.
* Please enter a valid value or be more precise than that, of course, and add NGF to conditionally show this, if the input value is wrong.
* Well, this gives us one additional problem though.
* How do we determine whether this input here or this control here does hold a wrong value? We'll take a closer look in the next lecture.