* -: Our form has evolved quite a bit.
* Now let's say that on the value object we get when we submit the form***, we want to group some things.***
* For example we want to group the secret and the question answer into username and email to just have some structure in our object because for a very big form, we might want to have such a structure.
* *This would also be nice if we could then validate the validity, the status of our individual groups of inputs.*
* Turns out that's easy to do with the template-driven approach.
* Here on our first group username and email I already have a wrapping div with the ID user data here.
* **ngModelGroup directive:**
* Now you can simply place another directive on it the ngModelGroup directive like this and this will now group this into, well you guessed it, a group of inputs.

Text

Description automatically generated

* However, ngModel group needs to be set equal to a string.
* So for example, user data this will be the key name for this group.
* So now if I save this with ngModel group added if I enter value here and here and hit submit and we have a look at the value of the form you now see that we have a user data field here which holds another object where we now have email and username.

Graphical user interface, application

Description automatically generated

* Now, not only did we add this extra field in our value we also now have a different setup here in controls.

Graphical user interface, application

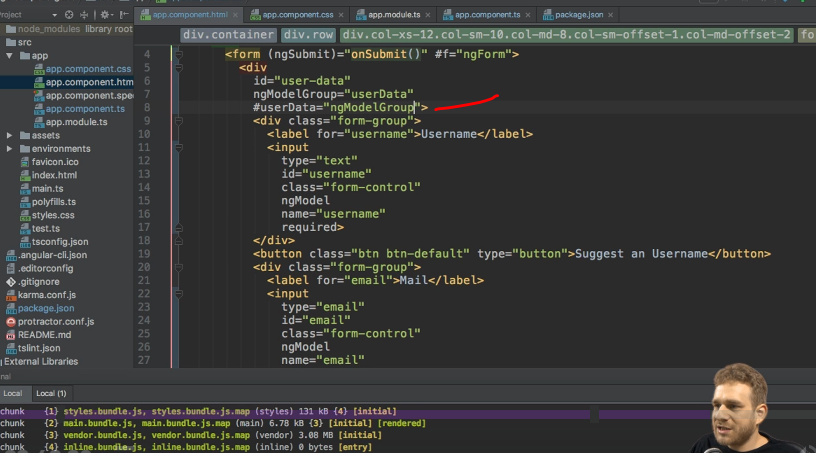
Description automatically generated

* Here we also now have a user data control with all the properties, you know on those controls like valid and so on.
* So if we now simply inspect our HTML code and this div with the ID user data, you see that there also we got the ng-dirty, touched and valid clauses added.

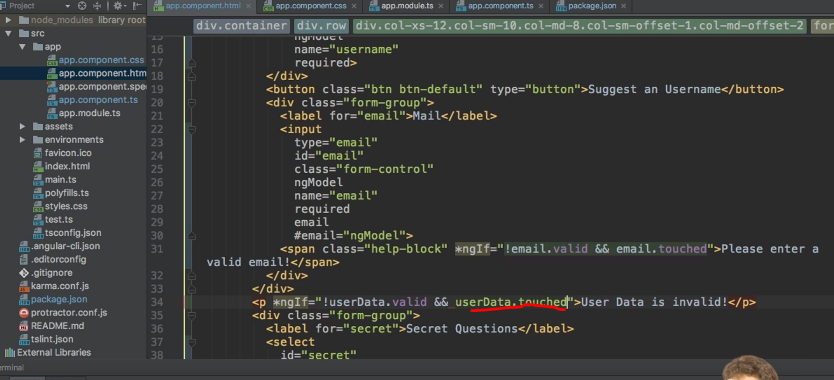
Graphical user interface, text, application

Description automatically generated

* So you can now also check the validity of this overall control here, for example which might be a nice feature in your form.
* ***You can also get access to the Java script representation by again adding a local rough reference to the element which holds the ngModel group directive.***
* Here, for example, user data would be a fitting name and then setting this equal to ngModel group.



* So just like we did before with email, which was equal to ngModel, I'm now setting this reference equal to ngModel group to get access to this JavaScript object.
* And this would allow us to, for example, output a message if this whole group is not valid.



* So we could simply output a paragraph here.
* User data is invalid which we add if userData, this is the local reference created here.
* If userData is not valid and let's say it has been touched.
* So userData.
* touched is true.

Graphical user interface, text, application, email

Description automatically generated

* With that, if this reloads, you see that we don't see any message but as soon as I click into one of the fields and then leave it, you see user data as invalid was shown.
* So now you really got a very fine grain control over your form with all these tools.