* In the last lecture, we created our form here in TypeScript.
* The issue is, it's nice to have it here, but our actual form, you could say, lives in the HTML template.
* So we somehow need to **synchronize our HTML inputs and our own form**, or this form in general with our own form.
* Right now, Angular doesn't know which of our TypeScript controls here relates to which input in our template code.
* It actually doesn't even know that our form, signup form here, should be attached to this form.
* Right now, the one thing it is doing is, it is auto-detecting that this is a form and it creates a form for us.
* Now we don't want it to do that, so we have to add some directives to override this default behavior to give Angular different instructions.
* For these directives to work, you definitely need to make sure that you add the import to the reactive forms module in your app module, otherwise you will get errors.
* **A)FormGroup Directive**
* Now, the first directive we need to add is here on the form itself, via property binding, the form group directive.
* Now this simply tells Angular, "Hey please take my form group.
* Don't infer one.

Text

Description automatically generated

* Don't create a form for me.
* Use my form group.

Text

Description automatically generated

* " And we need to set up property binding here, because we need to pass our form as an argument to the directive.
* So here we should reference our signup form, the property we created here, which stores our form.
* We're passing this via property binding to the form group, and ***now this form is actually synchronized with the form we created in TypeScript.***
* But we still need to *tell Angular which controls should be connected to which inputs in the template code.*
* For this, we get a number directive**.**
* **B)FormControlName directive:**
* On this input here, for example, the username, we add the form control name directive to tell Angular, "hey what's the name of this input in my form, in my TypeScript form?"

Text

Description automatically generated

* Well, the name is "username," this name here, because that's the control I want to connect to this input.

Text

Description automatically generated

* So I simply pass username here.
* If you're wondering why I'm not using property binding, I'm passing a string here.
* So if you want to use property binding, you can do this by wrapping this in square brackets and then enclosing username in single quotation marks.
* Otherwise, it would search for a property named username.
* But this is overly complicated.
* If you just want to pass a string, simply omit the square brackets and you're good to go.
* So if this were telling Angular*, "Hey, my form should be connected to the form stored in the sign-up form property, and in this form, this input here, should be connected to the control with the name username.*
* " Well, now I can simply repeat this for the email, here the name was "email," the name we set up here.
* Now this control is connected and for the gender, of course.

Text

Description automatically generated

* So in this input here, I will bind this to gender.
* Now if this.
* .
* .
* let's save this, so that it compiles, and now let's have a look at it.

Graphical user interface, text, application, email

Description automatically generated

* We don't see any errors, which looks good.
* We see that mail is pre-selected.
* Looks promising, because that was the default value we set when we created the control in TypeScript.
* And if I now switch to female, only one of the two is selected.

Graphical user interface, text, application

Description automatically generated

* We also see that if we select email here that the Angular classes are attached, untouched, pristine, valid, and so on.
* So it indeed looks like our inputs in HTML are correctly synchronized to our form we created in TypeScript.
* So this was successful.
* The next step is to actually submit the form to see how this works.