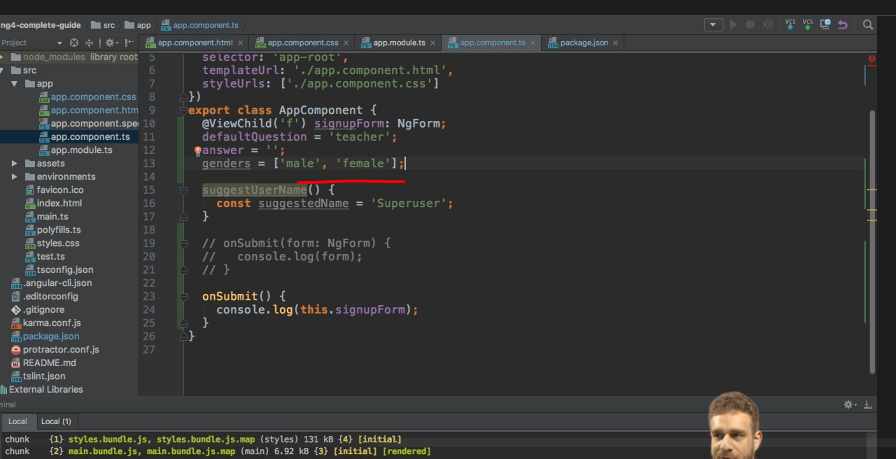
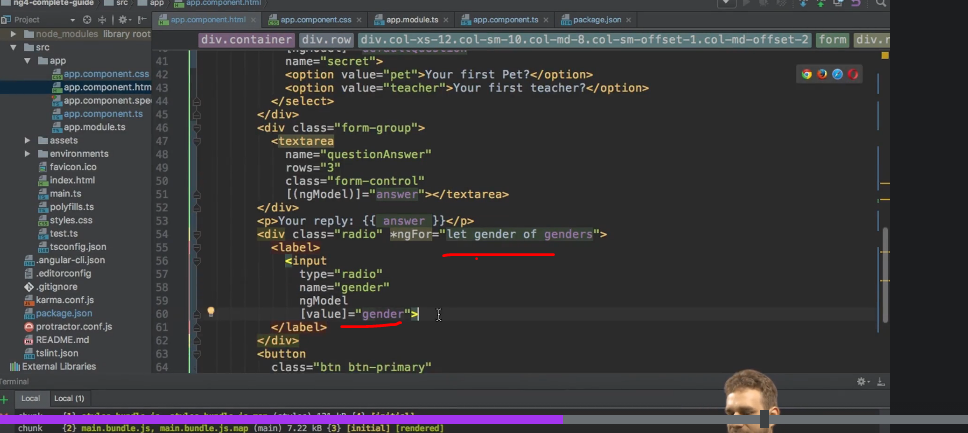
* -: We're nearing the end of the first part of this section, the template-driven approach.
* There's still some things I want to show you.
* For example, how to work with radio buttons.
* We don't have any radio buttons in our form as of now, so let's add some.
* Let's add a property in the app component first with some genders.
* So this is simply an array where I have male and female, and I don't want to insult anyone, I know there are more genders nowadays.



* I'm going to go with the classic approach here.
* Just to keep it simple.
* We have this genders array, and now with that array I want to output genders here.
* So let's say below our, your reply paragraph here, I'll add a new div with a class of radio.
* And this div should be replicated for all the genders.
* So I'm going to loop through my genders, maybe bind it to a gender variable, and this is just a set of bootstrap uses to give to give to some nice styling.
* Of course, use any other setup to style your radio buttons here.
* I'm looping through all the genders.



* And now again, in the bootstrap ball to make a, to create a nice looking radio button.
* I'm going to wrap it in a label.
* This input here, which is of type radio.
* And I'm going to wrap this over multiple lines too just to make it all super simple to read and to understand.
* And here I will give it a name of gender because on a radio button, I can only select one of them in a group.
* So the name should be gender for whichever button is selected.
* And I will place ngModel on it to make it a control.
* And I will pre-populate or I will set the value of this button equal to gender.
* So this variable of my ng4 loop.
* Now with that, if we save this, we should see that we have some gender buttons.
* To output a text, I need to add it here after my input.
* So again, gender here, like that.
* And now we have male and female here.
* And if I just enter something so that I can submit the form and I pick male here, you will see that on the value object here we got gender, which is male.

Graphical user interface, text, application

Description automatically generated

* And of course, if I pick female, this will, you guessed it, be set to female.
* So this is how you can easily incorporate radio buttons by binding or by placing ngModel on the input as always.
* Setting the value, of course, you could also hard code the value and that, well, makes it usable just like any other input.
* And if you want to set it to a default gender, of course, you can use one way binding again to make sure that one of the two buttons is selected by default at the start when you load the form.
* You can also, of course, add the required directive or attribute to that input here to make sure that now the form will not be valid until one of the two has been selected.
* So that works just like on any other input.
* So I just wanted to highlight this because radio buttons can look like a very special case.
* They aren't.
* They are used like any other input when using the template-driven approach in your Angular app.