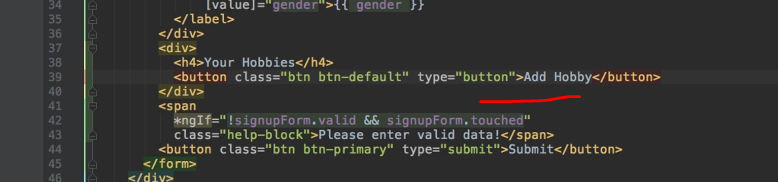
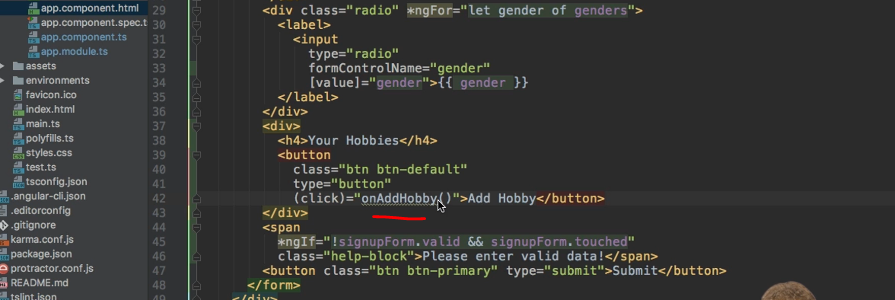
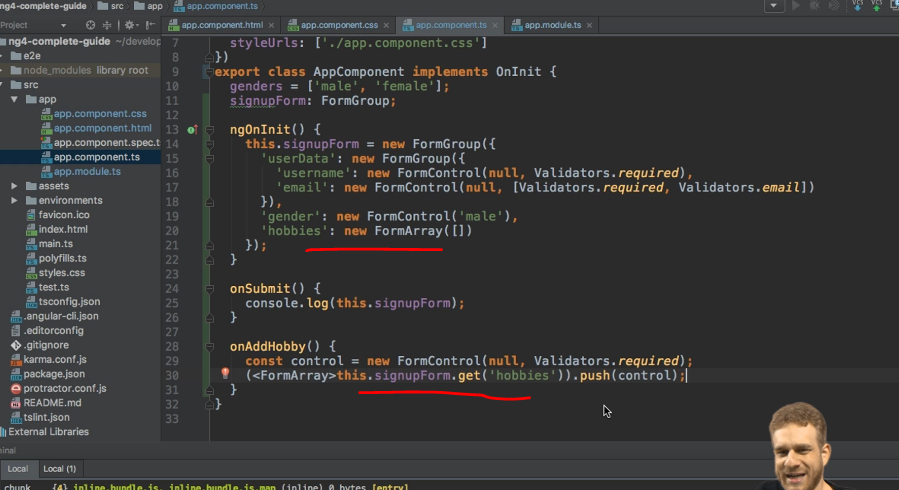
* -: We covered a lot about the reactive approach.
* There is one other nice feature I want to show you though.
* Let's add a new area to our form, maybe below the radio buttons here.
* Here I want to add a new div.
* *And in this div want to allow the user to dynamically add form controls.*
* Let's say, I want to allow the user to add his hobbies so Your Hobbies, something like that.
* Then I'll add a button with some bootstrap classes button and then maybe button default.
* Make it of type button.
* That's important so that this button doesn't accidentally submit the whole form.
* And here I'll say Add Hobby.



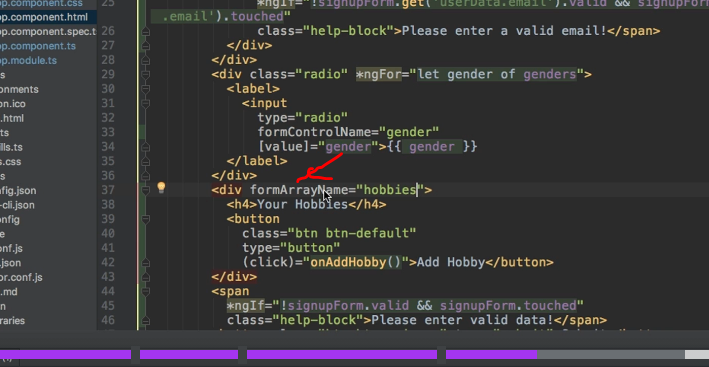
* Now when a user clicks the button I want to dynamically add a control to my form here.
* And that's actually super simple to do.
* Specifically I want to add this control to an array of controls because I might have multiple hobbies.
* So I'll add a click listener here on onAddHobby.
* Seems like a, like an appropriate name for this method here.
* And in there again, I want to add controls.



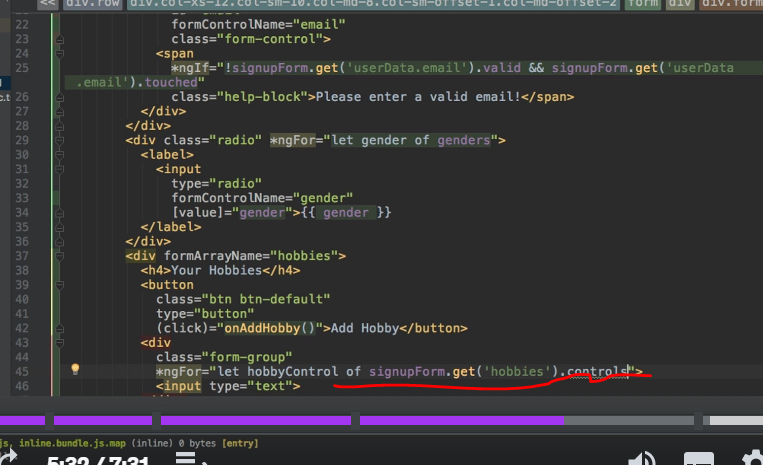
* So I'll add the method in my types group code onAddHobby, like this.
* And now I need to add a new control, type of control to my overall form.
* I'll name it hobbies, because that is what I'll store in there in the end.
* And as I said, this should be an array because it could be no hobbies, or 10, or 100.
* So the type here is not form control and it is not form group.
* Instead it's a **FormArray**.
* Make sure to import this from @angular/forms as well.
* Now a ***FormArray, of course, holds an array of controls***.
* So you pass an array here to initialize it.
* In this array you could already initialize some form controls with new FormControl or like I am doing it here, you leave it empty to not have any hobbies at the beginning.
* Now when we click onAddHobby I want to add a new hobby to that array.
* So here I need to access my form.
* Now I want to get access to my hobbies array.
* And I can do this by getting hobbies.
* It is a control in my overall form in the end.
* And now I need to tell TypeScript that this is of type form array to not get an error.
* You rarely have to do this but here we have to explicitly cast this.
* So by placing a lower than sign than FormArrays of the type and then a greater than sign and then placing this in parentheses we're telling TypeScript, This part here actually is a form array, so everything enclosed in these outer parenthesis now is treated as form array.
* So now I can push a new control on this array.
* If it would've not casted this, we would get an error.
* So now we can push a new form control there but I will outsource this to keep those lines short.
* So here the control is created and stored in a constant, FormControl.
* And the hobby, well that should be something the user can now enter.
* So we'll actually create it with no default value.
* Of course you could all change this behavior to pass an argument to onAddHobby and then pre-populate it with this value.
* I want to add a validator though.
* So this still works, because I'm creating a normal form control here.
* So the required validator seems appropriate because the hobby input should not be empty.
* And now I can push this control on my array of controls.



* With this we are creating, as we are able to add controls, ***but we won't see it.***
* ***We need to synchronize it with our HTML code.***
* Therefore, on this outer div here, I'll add a directive **form ArrayName.**
* So again, you see the pattern.
* We had form ControlName to tell Angular that we want to connect the input to a FormControl.
* We had formGroupName to do the same for a form group.
* And now we have formArrayName to do the same for an array.
* So formArrayName and the name was hobbies.
* The name we chose in our TypeScript object.



* This tells Angular that somewhere in this div our array will live.
* Now we can use this array here.
* So I will create a new div with the ID form-group, for example, bootstrap styling.
* And here I simply want to have an input which allows the user to enter the actual hobby.
* So this type text is all right.
* I now somehow need to loop through all the controls which are in this array.
* So on my form-group here will actually add a ngFor loop to loop through all my hobby controls.
* So in my signupForm, that would be my hobbies, which I can get like this.
* And there're the controls.



* With that, I can loop through all them.
* And I also want to extract the index of the current iteration.
* I will need this to assign this input to one of these dynamically created controls.
* Because on this input, I of course want to add the form-control CSS class.
* But besides this, very important I need to add formControlName, 'cause we still need to synchronize this input with the dynamically created input.
* Now, this dynamically created input will not have a name chosen by us, but it isn't an array.
* So the name will simply be the index in this array which is why I'm retrieving it here.

Text

Description automatically generated

* So I can simply bind form ControlName and now I will need property binding because I'm now not passing a string, I'm instead passing this local variable I'm creating here, i, to pass this index.

Text

Description automatically generated

* Let's save this and see if this works as expected.
* If we have a look at our console, no errors.
* Let's click add hobby, and we indeed see a new input.
* No errors still.
* Let's check it.
* It's invalid, this looks good.

Graphical user interface, text, application, email

Description automatically generated

* So if I enter a hobby here, now it becomes valid.
* And now if I enter the rest here and go to the console and hit submit, you will see that in the value of this form, we actually now have hobbies which is an array where we have cooking, and of course we could also add sports.

Graphical user interface

Description automatically generated

* And this newly submitted form, you would now find that this array, of course, holds cooking and sports.
* So this is how you can use a form array with your reactive approach very easily.
* It's all about synchronizing it in the end and about knowing that you have to explicitly cast it to make this work.
* Maximilian posted an announcement · 6 months ago ·