ANGULAR FORMS

GENERAL

- ❑ You can simply submit to the server but then keep in mind, you're creating a single page application, so there is no submitting to the server,
- ❑ Instead you will need to handle the form through Angular and if you then want to submit something to the server, you will need to reach out via Angular's HTTP

Forms in angular

- ➔ Angular's job now is to allow you to retrieve the values the user entered here and also to check some other things, like is the form valid, did the user enter valid information?
- ➔ So you somehow need to be able to parse the values the user enters and you somehow need some Javascript object representation of your form in your TypeScript code to work with, so this object could look something like this.
- ➔ It is more complex in reality but these would be the key features we need for example, we need to have the value of the form and there it would be convenient if we would have key-value pairs where the key refers to the name of an input.
- ➔ For example name or email as specified in the HTML code and then it holds the values the user

- ➔ Entered, this would make it super simple for you to work with these values in your TypeScript code and to do something with the values the user entered.
- ➔ It might also be helpful if some metadata was stored, like for example if the form is valid and this would be changed to false.
- ➔ For example if the e-mail address wouldn't be a valid e-mail address. This is what Angular does for you, it allows you or it gives you actually such a Javascript object representation of your form, making it simple for you to retrieve user values and to see the state of the form and to work with it.

Example :

```
{
value:
{
  name : ‘Max’,
  email : test@gmail.com
}
valid : True
}
```

Template Driven vs Reactive forms

TD - Angular infers form object from DOM

RF- Form is created programmatically and synchronized with DOM

Example :

```
Import { FormsModule } from ‘@angular/forms’;
Imports :
[
FormsModule
]
```

.html

```
<form (ngSubmit) = “onSubmit(f)” #f=”ngForm”>
<input  type=”email” id=”email” class=”form-control”
ngModel name=”secret”>
```

.ts

```
onSubmit(form : ngForm)
{
console.log(‘Submitted’);
}
```

Accessing the form using ngChild

@ViewChild('f') signupForm : NgForm;

onSubmit(form : ngForm)

{

console.log(this.signupForm);

}

Form validation

Add required attribute => acts as a selector

Which Validators do ship with Angular?

Checkout the Validators class: https:// angular.io /api /forms / Validators - these are all built-in validators, though that are the methods which actually get executed (and which you later can add when using the reactive approach).

For the template-driven approach, you need the directives. You can find out their names, by searching for "validator" in the official docs: https://angular.io/api?type=directive - everything marked with "D" is a directive and can be added to your template.

Builtin Angular form validation using HTML 5

If not valid disable the button

Eg :

<button class ="btn btn-primary" type="submit"

[disabled]="!f.valid">

.css

```
.ng-invalid {
border : 1px solid red;
}
input.ng-invalid {
border : 1px solid green;
}
```

Output validation error msg :

- ❏ If e-mail is not valid, so add an exclamation mark at the beginning. Now with this if this reloads, you'll see that help text and of course you can improve this by also chaining another condition that e-mail should have been touched to give the user a chance of changing it.
- ❏ So now you don't see the warning but if you click in there and click out of there and it is invalid, you see that warning and if you now enter a valid data, it disappears.
- ❏ So this is another way of taking advantage of the state managed by Angular
- ❏ but this also shows you how you can get a quick and easy access to control added by Angular.
- ❏ And with that, you should have tools to really provide a pleasant user experience showing the right errors, the right warnings and styling the form correctly depending on the state of the form.

Example

```
<input  type=”email” id=”email” class=”form-control”
ngModel name=”secret”
required
email
#email=”ngModel”>
<span class=”help-block”  *ngIf=”!email.valid && email.touched”>
</span>
```

Set default values with ngModel Property Binding

For example on the dropdown here. Right now, it's empty by default, well would be nice if one of the two questions would be selected, right? For this, we can change the way we add ngModel.

```
<select
id=”secret” class=”form-control”
[ngModel]=”defaultQuestion”
ngModel name=”secret”>
<option value=”pet”>Your 1st pet ? </option>
<option value=”teacher”>Your favourite teacher ? </option>
```

.ts

```
defaultQuestion=’pet’;
```

Using ngModel with 2 way binding

- ❖ We don't use two-way binding, ngModel is either used without any binding or with one-way binding
- ❖ Sometimes you not only want to pre-populate a default the user entered but you also want to instantly react to any changes.
- ❖ But sometimes you instantly want to check something or simply repeat whatever the user entered

Example :

```
 <div class = “form-group”>
        <textarea
                name = “questionAnswer”
                rows=”3”
                [(ngModel)]=”answer”>
        </textarea>
<.div>
<p> Your reply : {{answer}} </p>
```

.ts:

```
answer=”;
```

Handling radio buttons

```
<div class=”radio” *ngFor=”let gender of genders”>
```

```
        <label>
                <input type=”radio”
                name=”gender”
                ngModel [value]=”gender”>
        </label>
</div>
```

.ts

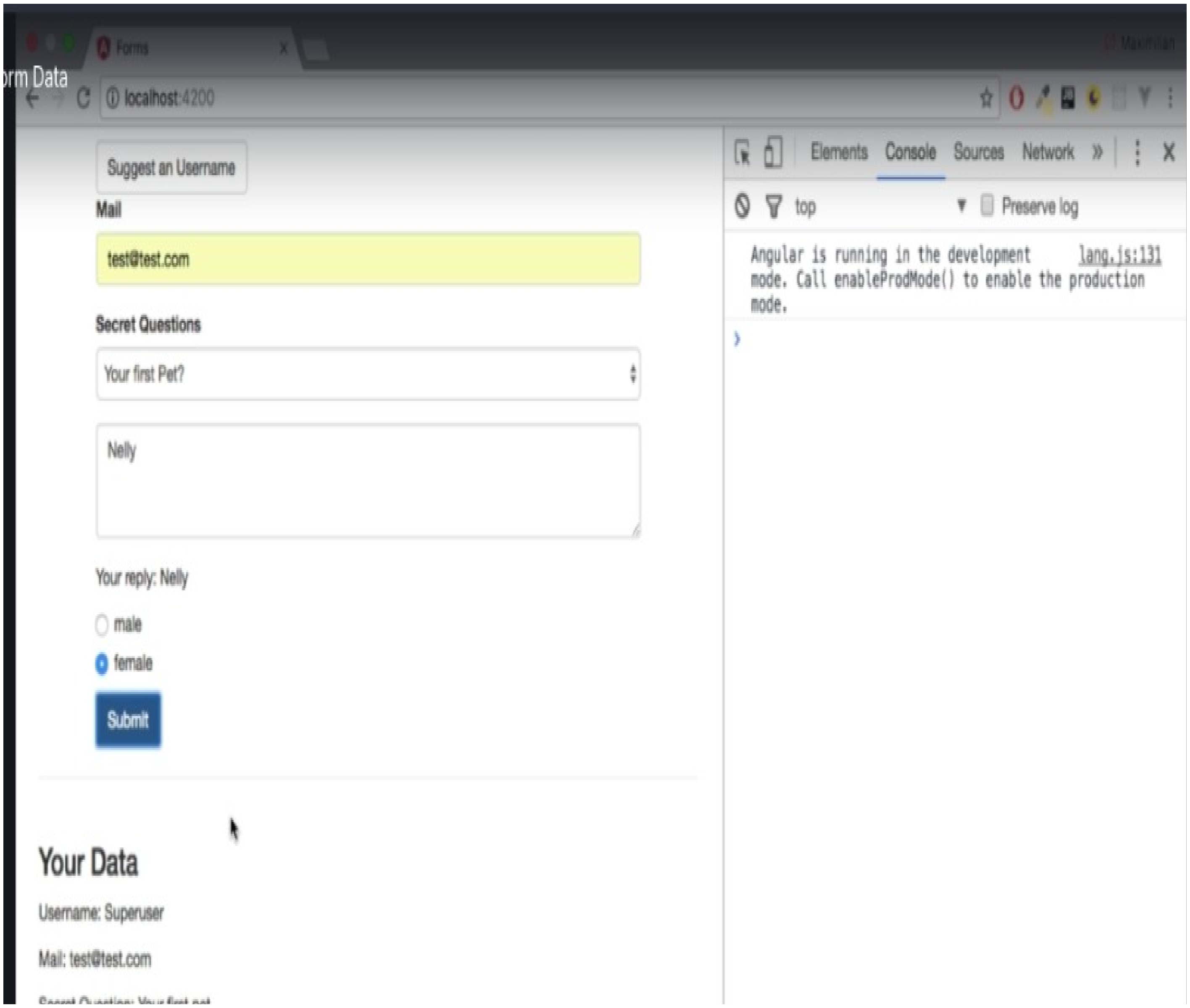
```
genders=[‘male’,’female’]
```

Using form data

```
@ViewChild(‘f’) signupForm : NgForm;
user =
{
        username:”,
        email:”,
        secretQuestion:”,
        answer:”,
        gender:”
};
onSubmit()
{
        this.user.username = this.signupForm.value.userName;
}
```

RESETTING THE FORM

this.signupForm.resubmit()



```
Angular is running in the development     lang.js:131
mode. Call enableProdMode() to enable the production
mode.
```

Mail: test@test.com

Secret Questions: Your first Pet?

Nelly

Your reply: Nelly

( ) male
(●) female

Your Data

Username: Superuser

Mail: test@test.com