**Async form validation in Angular**

**Setup**

The idea is to validate a form’s input field against server fetched data on the spot, before submitting anything. For a simple example, server will return a list of letters. Letters must be unique, so if a letter exists, we get validation error.

There’s no reason to bother with an actual backend, we’ll just mock an http call for data fetching:

export function fetchLetters() {  
 return new Observable<string[]>(sub => {  
 const t = setTimeout(() => {  
 sub.next(['a', 'b', 'c']);  
 sub.complete();  
 clearTimeout(t);  
 }, 1000)  
 })  
}

This is what Angular’s HttpClient function would normally return, with a dummy delay of one second.

Moving on to bare minimum html:

<form [formGroup]="form" (submit)="onSubmit($event)">  
 <label>  
 Letter: <input formControlName="letter" />  
 </label>  
 <div \*ngIf="form.controls.letter.errors?.['valueError']">  
 {{ form.controls.letter.errors?.['valueError'] }}  
 </div>  
 <button type="submit">Submit</button>  
</form>

Form, label, input, button and an error div… Standard stuff. Make sure to import ReactiveFormsModule in your component .ts file:

@Component({  
 // ... boiler  
 imports: [CommonModule, ReactiveFormsModule],  
 // ... plate  
})  
export class Dumbass...

And time to implement FormGroup that will be binded to html:

form = new FormGroup({  
 letter: new FormControl('', {  
 validators: [Validators.required, Validators.minLength(1), Validators.maxLength(1)],  
 asyncValidators: [this.letterValidator()],  
 updateOn: 'submit' // 'blur' | 'change'  
 })  
});

So there’s a form control with both validators and asyncValidators defined. If regular validators fail, async validators won’t be called, which is nice. No need to bombard the server with useless requests. updateOn property controls when the validation is triggered.

letterValidator() is a function that must return an object which implements AsyncValidatorFn interface. It’s just a contract for how form validators must be implemented in Angular.

Form validator function takes an AbstractControl object (which FormControl implements), and it must return nullif validation has passed, or a dictionary with error keys and descriptions if not. Angular aliased it under ValidationErrors type.

**Async**form validator is the same, but it must wrap the return type under Observable or Promise for… obvious reasons…

I still think using promises is extremely stupid in Angular, so I’ll go with observables:

letterValidator(): AsyncValidatorFn {  
 return (control: AbstractControl) => {  
 return fetchLetters()  
 .pipe(  
 map(x =>  
 // actual validation logic here  
 x.includes(control.value)  
 ? { 'valueError': 'value already exists' } as ValidationErrors  
 : null)  
 )  
 }  
 }

Just return a function that calls fetchLetters() function (or any function that returns required data from the server), and in the map() do the actual validation part. It’s useful to annotate the types explicitly. I’ve only defined valueError here, but you can theoretically add as much things as you like.

Finally, for the completion sake, the onSubmit() function:

onSubmit(ev: Event) {  
 ev.preventDefault();  
 // handle submit...  
 }

**Conclusion?**

Fetch the goddamn data before loading the form, and just use normal validators.