Angular - Dynamically add/remove validators

Asked 1 year, 7 months ago Active 6 months ago Viewed 17k times

13

3

I have a `FormGroup` defined like below:

```
this.businessFormGroup: this.fb.group({
    'businessType': ['', Validators.required],
    'description': ['', Validators.compose([Validators.required,
Validators.maxLength(200)])],
    'income': ['']
  })
```

Now when `businessType` is `Other`, I want to remove `Validators.required` validator from `description`. And if `businessType` is not `Other`, I want to add back the `Validators.required`.

I am using the below code to dynamically add/remove the `Validators.required`. However, it clears the existing `Validators.maxLength` validator.

```
if(this.businessFormGroup.get('businessType').value !== 'Other'){
    this.businessFormGroup.get('description').validator =
<any>Validators.compose([Validators.required]);
} else {
    this.businessFormGroup.get('description').clearValidators();
}

this.businessFormGroup.get('description').updateValueAndValidity();
```

My question is, how can I retain the existing validators when adding/removing the `required` validator.

angular angular-forms angular-validation

asked Mar 2 '18 at 18:13

A J Qarshi
1,140 2 24 48

sadly this is not possible, angular seems to merge the applied validators internally therefore you can only call `clear` and `set` functions – Nickolaus Mar 2 '18 at 19:28

@Ricardo thats wrong, validators are composed into a single function and thats it. With the current implementation of the API its not possible to check which validators are set for a control – Jota.Toledo Mar 2 '18 at 20:24

4 Answers

21

Angular forms have a built in function setValidators() that enables programmatic assignment of Validators.

For your example you can do:

```
if(this.businessFormGroup.get('businessType').value !== 'Other'){
    this.businessFormGroup.controls['description'].setValidators([Validators.required, 
Validators.maxLength(200)]);
} else {

this.businessFormGroup.controls['description'].setValidators([Validators.maxLength(200)]);

}
```

It is important to keep in mind that **by using this method you will overwrite your existing validators** so you will need to include all the validators you need/want for the control that you are resetting.

edited Oct 16 '18 at 19:51

answered Mar 2 '18 at 19:32
Narm
4,520 1 20 42

2 note that the setValidators will overwrite previously setted validators, meaning that the maxLength will be overwritten and wont be regenerated with this approach – Jota.Toledo Mar 2 '18 at 19:36

Thanks @Jota.Toledo. That is an important behavior to be aware of. Updated the answer to include it. – Narm Mar 2 '18 at 19:50

I think that use a customValidator that take account of bussinesType and description is a "more natural" idea that remove/add validators – Eliseo Mar 2 '18 at 21:08

worked for me !!!! – Sarjerao Ghadage May 13 at 9:27

4

This one work for me

```
    onAddValidationClick(){
          this.formGroup.controls["firstName"].setValidators(Validators.required);
         this.formGroup.controls["firstName"].updateValueAndValidity();
       }

 onRemoveValidationClick(){
          this.formGroup.controls["firstName"].clearValidators();
         this.formGroup.controls["firstName"].updateValueAndValidity();
       }
```

edited Mar 25 at 5:52

answered Mar 25 at 5:45
San Jaisy
3,571 10 52 92

3

The naive approach would be to set the validators of the control whenever the conditional variable changes. But we can actually do better than that by using some indirection + functional programming.

Consider the existence of a `descriptionIsRequired` getter, that acts as a boolan flag.

Ideas:

- Create a custom validator function that takes the `descriptionIsRequired` as argument and depending on it validates a control against required + maxLength or maxLength.
- Bind the custom validator to the description control in such a way, that when the validity of the control is evaluated, the newest value of `descriptionIsRequired` should be considered.

The first point is pretty straight forward to implement:

```
function descriptionValidator(required: boolean): ValidatorFn {
  return (formControl: FormControl): ValidationErrors => {
    } else {
      return Validators.maxLength(200)(formControl);
    }
  }
}
```

Notice that this is a self capsulated function.

The second point is a little bit more tricky, but in the end it looks like this:

```
export class FooComponent {
  constructor(){
    this.form = fb.group({
      description: ['initial name', this.validator()]
    });
  }

  private get descriptionIsRequired(): boolean {
   ...
  }

  private validator(): ValidatorFn {
    return (c: FormControl): ValidationErrors =>
descriptionValidator(this.descriptionIsRequired)(c);
  }
}
```

A small explanation of what is happening:

- the `validator` method returns a function
- the function returned by `validator` could be considered a *factory method*: whenever its invoked, returns a new function, more specifically, a new instance of our `descriptionValidator` using the newest `descriptionIsRequired` value.

A live demo in the following stackblitz

edited Mar 3 '18 at 8:56

answered Mar 2 '18 at 21:37
Jota.Toledo
15.5k 7 38 56

+1 It's valid solution but I am looking for a more generic implementation as my validation rules are defined in a json file. – A.J.Qarshi Mar 2 '18 at

I wonder what do you mean by generic in this context. The only way I see for this to be generalized is that you have a collection of validators `V` that always have to be active in a control. Where they come from is irrelevant. Then you have another collection of validators `Vop`, in this case a collection with only required in it, that can and can not be active in a control depending on a boolean condition. Am I right? If no, please further explain what is your concept of "more generic" implementation. – Jota.Toledo Mar 3 '18 at 15:14

cheers! this gave me a good idea to do something generic I needed to do – Juan Stoppa Sep 13 '18 at 22:05

2

Maybe this helps:

Adding Validators.required to the validatorset of an existing `AbstractControl`:

```
if (c.validator !== null) {
        c.setValidators([c.validator, Validators.required])
  } else {
        c.setValidators([Validators.required])
  }
```

edited Feb 22 at 3:57
nircraft
4,117 3 15 32

answered Apr 21 '18 at 15:38
Rob
21 1