

# ValueChanges in Angular Forms

7 Comments / 7 minutes of reading / March 9, 2023

← [StatusChanges](#)

[Angular Tutorial](#)

[FormControl](#) →

The ValueChanges is an event raised by the [Angular forms](#) whenever the value of the [FormControl](#), [FormGroup](#), or [FormArray](#) changes. It returns an [observable](#) so that you can subscribe to it. The observable get the latest value of the control. It allows us to track changes made to the value in real-time and respond to them. For example, we can use it to validate the value, calculate the computed fields, etc.

## Table of Contents

[How to use ValueChanges](#)

[ValueChanges Example](#)

[ValueChanges of FormControl](#)

[ValueChanges shows the previous value](#)

[ValueChanges of FormGroup](#)

[ValueChanges of Form](#)

[EmitEvent & ValueChanges](#)

[OnlySelf & ValueChanges](#)

[Complete Source Code](#)

[ValueChanges in Template Driven Forms](#)

[Summary](#)

## How to use ValueChanges

[Angular Forms](#) has three building blocks. [FormControl](#), [FormGroup](#) & [FormArray](#). All of these controls extend the AbstractControl base class. The AbstractControl base class implements the ValueChanges event

We can subscribe to ValueChanges by getting the reference of the control and subscribing to it as shown below

```
1
2 this.reactiveForm.get("firstname").valueChanges.subscribe(x => {
3   console.log('firstname value changed')
4   console.log(x)
5 })
6
```

You can also subscribe to the top-level form as shown below.

```
1
2 this.reactiveForm.valueChanges.subscribe(x => {
3   console.log('form value changed')
4   console.log(x)
5 })
6
```

## ValueChanges Example

Create a [reactive form](#) as shown below

```
1
2 reactiveForm = new FormGroup({
3   firstname: new FormControl("", [Validators.required]),
4   lastname: new FormControl(),
5   address: new FormGroup{
6     city: new FormControl(),
7     street: new FormControl(),
8     pincode: new FormControl()
9   })
10 })
```

## ValueChanges of FormControl

You can subscribe to `ValueChanges` of a single `FormControl` as shown below. Here in `selectedValue` variable, we will get the latest value of the `firstname`. You can also retrieve the latest value of the `firstname` using `this.reactiveForm.get("firstname").value`

```
1
2 this.reactiveForm.get("firstname").valueChanges.subscribe(selectedValue => {
3   console.log('firstname value changed')
4   console.log(selectedValue)           //latest value of firstname
5   console.log(this.reactiveForm.get("firstname").value) //latest value of firstname
6 })
7
```

## ValueChanges shows the previous value

But, the top-level form is not yet updated at this point, hence `this.reactiveForm.value` still shows the previous value of the `firstname`.

The `valueChanges` event for the `firstname` fires immediately **after** the new values are updated but **before** the change is bubbled up to its parent. Hence the `this.reactiveForm.value` still shows the previous value.

```
1
2 this.reactiveForm.get("firstname").valueChanges.subscribe(selectedValue => {
3   console.log('firstname value changed')
4   console.log(selectedValue)
5   console.log(this.reactiveForm.get("firstname").value)
6   console.log(this.reactiveForm.value)  //still shows the old first name
7 })
8
```

You can work around this by waiting for the next tick using `setTimeout` as shown below.

```
1
2 this.reactiveForm.get("firstname").valueChanges.subscribe(selectedValue => {
3   console.log('firstname value changed')
4   console.log(selectedValue)
5   console.log(this.reactiveForm.get("firstname").value)
6   console.log(this.reactiveForm.value)  //shows the old first name
7
8   setTimeout(() => {
9     console.log(this.reactiveForm.value)  //shows the latest first name
10  })
11
12 })
13
```

## ValueChanges of FormGroup

The `ValueChanges` event of `FormGroup` or `FormArray` is fired, whenever the value of any of its child controls value changes. For Example, the following `ValueChanges` will fire even whenever the value of the *city*, *state* & *Pincode* changes.

```
1
2 this.reactiveForm.get("address").valueChanges.subscribe(selectedValue => {
3   console.log('address changed')
4   console.log(selectedValue)
5 })
6
```

## ValueChanges of Form

The following example show we can subscribe to the changes made to the entire form.

```
1  
2 this.reactiveForm.valueChanges.subscribe(selectedValue => {  
3   console.log('form value changed')  
4   console.log(selectedValue)  
5 })  
6
```

## EmitEvent & ValueChanges

The ValueChanges event is fired even when the values of the control are changed programmatically. In some circumstances, you might not want to raise the ValueChanges event. To do that we can use the emitEvent: false

In the following example, the ValueChanges event is **not fired** at all, even though the value of the firstname is changed.

```
1  
2 this.reactiveForm.get("firstname").setValue("", { emitEvent: false });  
3
```

You can use `emitEvent: false` with the [setValue](#), [patchValue](#), [markAsPending](#), [disable](#), [enable](#), [updateValueAndValidity](#) & [setErrors](#) methods.

## OnlySelf & ValueChanges

When `onlySelf: true` the changes will only affect only this `FormControl` and change is **not** bubbled up to its parent. Hence the `ValueChanges` event of the parent `FormGroup` does not fire.

For Example, the following code will result in the `ValueChanges` of the `firstname`. but not of its parent (i.e. top-level form)

```
1  
2 this.reactiveForm.get("firstname").setValue("", { onlySelf: true });  
3
```

You can use the `onlySelf: true` with the `setValue`, `patchValue`, `markAsUntouched`, `markAsDirty`, `markAsPristine`, `markAsPending`, `disable`, `enable`, and `updateValueAndValidity` methods

## Complete Source Code

[tabby title="reactive.component.ts"]

```
1  
2 import { Component, OnInit } from '@angular/core';  
3 import { FormGroup, FormControl, Validators } from '@angular/forms'  
4 import { timeout } from 'q';  
5  
6  
7 @Component({  
8   templateUrl: './reactive.component.html',  
9 })  
10 export class ReactiveComponent implements OnInit {  
11   title = 'Reactive Forms';
```

```
12
13 reactiveForm = new FormGroup({
14   firstname: new FormControl("", [Validators.required]),
15   lastname: new FormControl(),
16   address: new FormGroup({
17     city: new FormControl(),
18     street: new FormControl(),
19     pincode: new FormControl()
20   })
21 })
22
23 onSubmit() {
24   console.log(this.reactiveForm.value);
25 }
26
27 ngOnInit() {
28
29   this.reactiveForm.get("firstname").valueChanges.subscribe(selectedValue => {
30     console.log('firstname value changed')
31     console.log(selectedValue)
32     console.log(this.reactiveForm.get("firstname").value)
33     console.log(this.reactiveForm.value)
34
35     setTimeout(() => {
36       console.log(this.reactiveForm.value)
37     })
38
39   })
40
41   this.reactiveForm.get("address").valueChanges.subscribe(selectedValue => {
42     console.log('address changed')
43     console.log(selectedValue)
44   })
45
46   this.reactiveForm.valueChanges.subscribe(selectedValue => {
47     console.log('form value changed')
48     console.log(selectedValue)
49   })
50 }
51
52
53
54 setValue() {
55
56   let contact = {
57     firstname: "Rahul",
58     lastname: "Dravid",
59     address: {
60       city: "Bangalore",
```

```
61     street: "Brigade Road",
62     pincode: "600070"
63   }
64 };
65
66   this.reactiveForm.setValue(contact);
67 }
68
69   setAddress() {
70
71     this.reactiveForm.get("address").setValue(
72       {
73         city: "Bangalore",
74         street: "Brigade Road",
75         pincode: "600070"
76       }
77     );
78   }
79
80   setFirstname() {
81     this.reactiveForm.get("firstname").setValue("Saurav")
82   }
83
84   withoutOnlySelf() {
85     this.reactiveForm.get("firstname").setValue("");
86   }
87   withOnlySelf() {
88     this.reactiveForm.get("firstname").setValue("", { onlySelf: true });
89   }
90
91   withEmitEvent() {
92     this.reactiveForm.get("firstname").setValue("Sachin");
93   }
94   withoutEmitEvent() {
95     this.reactiveForm.get("firstname").setValue("", { emitEvent: false });
96   }
97
98   reset() {
99     this.reactiveForm.reset();
100   }
101
102 }
103
```

[tabby title="reactive.component.html"]



```
1
2 <h3>{{title}}</h3>
3
4 <div style="float: left; width:50%;">
5
6   <form [formGroup]="reactiveForm" (ngSubmit)="onSubmit()" novalidate>
7
8     <p>
9       <label for="firstname">First Name </label>
10      <input type="text" id="firstname" name="firstname" formControlName="firstname">
11      <label for="lastname">Last Name </label>
12      <input type="text" id="lastname" name="lastname" formControlName="lastname">
13    </p>
14
15    <div formGroupName="address">
16
17      <p>
18        <label for="city">City</label>
19        <input type="text" class="form-control" name="city" formControlName="city">
20        <label for="street">Street</label>
21        <input type="text" class="form-control" name="street" formControlName="street">
22        <label for="pincode">Pin Code</label>
23        <input type="text" class="form-control" name="pincode" formControlName="pincode">
24      </p>
25
26    </div>
27
28
29    <button>Submit</button>
30    <div>
31      <button type="button" (click)="setValue()">SetValue</button>
32      <button type="button" (click)="setAddress()">Address</button>
33      <button type="button" (click)="setFirstname()">First Name</button>
34    </div>
35    <div>
36      <button type="button" (click)="withoutOnlySelf()">Without Only Self</button>
```

```

37     <button type="button" (click)="withOnlySelf()">With Only Self</button>
38 </div>
39 <div>
40     <button type="button" (click)="withouEmitEvent()">Without EmitEvent</button>
41     <button type="button" (click)="withEmitEvent()">With EmitEvent</button>
42 </div>
43
44 </form>
45 </div>
46
47 <div style="float: right; width:50%;">
48
49     <h3>Form Status</h3>
50     <b>status : </b>{{reactiveForm.status}}
51     <b>valid : </b>{{reactiveForm.valid}}
52     <b>invalid : </b>{{reactiveForm.invalid}}
53     <b>touched : </b>{{reactiveForm.touched}}
54     <b>untouched : </b>{{reactiveForm.untouched}}
55     <b>pristine : </b>{{reactiveForm.pristine}}
56     <b>dirty : </b>{{reactiveForm.dirty}}
57     <b>disabled : </b>{{reactiveForm.disabled}}
58     <b>enabled : </b>{{reactiveForm.enabled}}
59
60
61     <h3>Form Value</h3>
62     {{reactiveForm.value |json}}
63
64 </div>
65

```

[tabbyending]

[tabby title="app.component.html"]

```

1
2 <h3>Angular ValueChanges Example</h3>
3
4 <ul>
5     <li>
6         <a [routerLink]="['/template']" routerLinkActive="router-link-active" >Template</a>
7     </li>
8     <li>
9         <a [routerLink]="['/reactive']" routerLinkActive="router-link-active" >Reactive</a>
10    </li>

```

```
11 </ul>
12
13 <router-outlet></router-outlet>
14
```

[tabby title="app.component.ts"]

```
1
2 import { Component } from '@angular/core';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent {
10 }
11
12
```

[tabby title="app.module.ts"]

```
1
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
5
6 import { AppRoutingModule } from './app-routing.module';
7 import { AppComponent } from './app.component';
8 import { TemplateComponent } from './template-component';
9 import { ReactiveComponent } from './reactive.component';
10
11 @NgModule({
12   declarations: [
13     AppComponent, TemplateComponent, ReactiveComponent
14   ],
15   imports: [
16     BrowserModule,
17     AppRoutingModule,
18     FormsModule,
19     ReactiveFormsModule
20   ],
21   providers: [],
```

```
22 bootstrap: [AppComponent]  
23 })  
24 export class AppModule { }  
25
```

[tabbyending]

## ValueChanges in Template Driven Forms

ValueChanges event can also be used in template-driven forms. All you need to do is to get the reference to the Form Model in the component as shown below.

```
1  
2 @ViewChild('templateForm',null) templateForm: NgForm;  
3
```

You can refer to the example code below

[tabby title="template-component.ts"]

```
1  
2 import { Component, ViewChild, ElementRef, OnInit, OnDestroy } from '@angular/core';  
3 import { NgForm } from '@angular/forms';  
4  
5  
6 @Component({
```

```
7   templateUrl: './template.component.html',
8 })
9 export class TemplateComponent implements OnInit {
10
11   title = 'Template driven forms';
12
13   @ViewChild('templateForm', null) templateForm: NgForm;
14
15   contact: contact;
16
17   onSubmit() {
18     console.log(this.templateForm.value);
19   }
20
21   ngOnInit() {
22
23     setTimeout(() => {
24
25       this.templateForm.control.get("firstname").valueChanges.subscribe(selectedValue => {
26         console.log('firstname value changed')
27         console.log(selectedValue)
28         console.log(this.templateForm.control.get("firstname").value)
29         console.log(this.templateForm.control.value)
30
31         setTimeout(() => {
32           console.log(this.templateForm.control.value)
33         })
34       })
35
36       this.templateForm.control.get("address").valueChanges.subscribe(selectedValue => {
37         console.log('address changed')
38         console.log(selectedValue)
39       })
40
41       this.templateForm.valueChanges.subscribe(selectedValue => {
42         console.log('form value changed')
43         console.log(selectedValue)
44       })
45
46     });
47
48   }
49
50
51   setValue() {
52     let contact = {
53       firstname: "Rahul",
54       lastname: "Dravid",
55       address: {
```

```
56     city: "Bangalore",
57     street: "Brigade Road",
58     pincode: "600070"
59   }
60 };
61
62   this.templateForm.setValue(contact);
63 }
64
65 setAddress() {
66   let address= {
67     city: "Bangalore",
68     street: "Brigade Road",
69     pincode: "600070"
70   };
71
72   this.templateForm.control.get("address").setValue(address);
73
74 };
75
76 setFirstname() {
77   this.templateForm.control.get("firstname").setValue("Saurav")
78 }
79
80
81 withoutOnlySelf() {
82   this.templateForm.control.get("firstname").setValue("");
83 }
84 withOnlySelf() {
85   this.templateForm.control.get("firstname").setValue("", { onlySelf: true });
86 }
87
88 withouEmitEvent() {
89   this.templateForm.control.get("firstname").setValue("Sachin");
90 }
91 withEmitEvent() {
92   this.templateForm.control.get("firstname").setValue("", { emitEvent: false });
93 }
94
95 reset() {
96   this.templateForm.reset();
97 }
98
99 }
100
101
102 export class contact {
103   firstname:string;
104   lastname:string;
```

```
105 gender:string;  
106 email:string;  
107 isMarried:boolean;  
108 country:string;  
109 address: {  
110   city:string;  
111   street:string;  
112   pincode:string;  
113 }  
114 }  
115 }
```

[tabby title="template-component.html"]

```
1  
2 <h3>{{title}}</h3>  
3  
4 <div style="float: left; width:50%;">  
5   <form #templateForm="ngForm" (ngSubmit)="onSubmit(templateForm)">  
6  
7     <p>  
8       <label for="firstname">First Name </label>  
9       <input type="text" id="firstname" name="firstname" #fname="ngModel" ngModel>  
10  
11     </p>  
12     <p>  
13       <label for="lastname">Last Name </label>  
14       <input type="text" id="lastname" name="lastname" ngModel>  
15     </p>  
16  
17     <div ngModelGroup="address">  
18  
19       <p>  
20         <label for="city">City</label>  
21         <input type="text" id="city" name="city" ngModel>  
22         <label for="street">Street</label>  
23         <input type="text" id="street" name="street" ngModel>  
24         <label for="pincode">Pin Code</label>  
25         <input type="text" id="pincode" name="pincode" ngModel>  
26       </p>  
27  
28     </div>  
29  
30     <button>Submit</button>  
31   </div>  
32   <button type="button" (click)="setValue()">SetValue</button>
```

```

33     <button type="button" (click)="setAddress()">Address</button>
34     <button type="button" (click)="setFirstname()">First Name</button>
35 </div>
36 <div>
37     <button type="button" (click)="withoutOnlySelf()">Without Only Self</button>
38     <button type="button" (click)="withOnlySelf()">With Only Self</button>
39 </div>
40 <div>
41     <button type="button" (click)="withouEmitEvent()">Without EmitEvent</button>
42     <button type="button" (click)="withEmitEvent()">With EmitEvent</button>
43 </div>
44
45 </form>
46 </div>
47
48 <div style="float: right; width:50%;">
49     <h3>Form Status</h3>
50     <b>status : </b>{{templateForm.status}}
51     <b>valid : </b>{{templateForm.valid}}
52     <b>invalid : </b>{{templateForm.invalid}}
53     <b>touched : </b>{{templateForm.touched}}
54     <b>untouched : </b>{{templateForm.untouched}}
55     <b>pristine : </b>{{templateForm.pristine}}
56     <b>dirty : </b>{{templateForm.dirty}}
57     <b>disabled : </b>{{templateForm.disabled}}
58     <b>enabled : </b>{{templateForm.enabled}}
59
60     <h3>Form Value</h3>
61     {{templateForm.value | json }}
62 </div>>
63

```

[tabbyending]

## Summary

In this tutorial, we learned how to make use of ValueChanges in Angular Forms. The ValueChanges event is fired whenever the value of the FormControl, FormGroup, or FormArray changes. It is observable and we can subscribe to it. We can then use it to validate the forms. update the computed fields, etc. The ValueChanges event does not fire depending on how we set emitEvent or onlySelf, when updating the value and validity of the form controls.