

# SetValue & PatchValue in Angular

4 Comments / 11 minutes of reading / March 9, 2023

← [FormBuilder](#)

[Angular Tutorial](#)

[StatusChanges](#) →

In this tutorial, we will learn how to set the model values in [Reactive Forms](#). It is done using the `setValue` and `patchValue` methods provided by the `AngularFormsModule`. In this post, we will learn more about `setValue` and `patchValue` and also learn the difference between them. We also learn about the `onlySelf` & `emitEvent` arguments with an example.

## Table of Contents

[Angular Forms](#)

[SetValue](#)

[PatchValue](#)

[onlySelf](#)

[emitEvent](#)

[SetValue Vs PatchValue](#)

[Example form setup](#)

[SetValue & PatchValue in Reactive Forms](#)

[Setting Initial /Default Value](#)

[Nested FormGroup](#)

[FormControl](#)

[PatchValue](#)

[Reset Form](#)

[OnlySelf Example](#)[emitEvent example](#)[SetValue & PatchValue in Template-driven Forms](#)[Summary](#)

## Angular Forms

The Angular has two ways to build the forms. One using the [Template-driven approach](#) & the other one is the [reactive forms approach](#). We covered both in our previous tutorial. The list of all tutorials on Angular forms is here.

1. [Angular Forms Tutorial: Fundamental & Concepts](#)
2. [Template Driven Forms in Angular](#)
3. [Set Value in Template Driven forms in Angular](#)
4. [Reactive Forms in Angular](#)
5. [FormBuilder in Reactive Forms](#)
6. [SetValue & PatchValue in Angular](#)
7. [StatusChanges in Angular Forms](#)
8. [ValueChanges in Angular Forms](#)
9. [FormControl](#)
10. [FormGroup](#)
11. [FormArray Example](#)
12. [Build Dynamic or Nested Forms using FormArray](#)
13. [Validations in Reactive Forms in Angular](#)
14. [Custom Validator in Reactive Forms](#)
15. [Passing Parameter to Custom Validator in Reactive Forms](#)
16. [Inject Service into Custom Validator](#)
17. [Validation in Template Driven Forms](#)
18. [Custom Validator in Template Driven Forms](#)

The [Angular Forms](#) has three main building blocks i.e FormControl , FormGroup & FormArray . All these components have methods setValue & patchValue and behave differently

## SetValue

```
setValue(value: { [key: string]: any; }, options: { onlySelf?: boolean; emitEvent?: boolean; } = {}): void
```

We use the SetValue to update the FormControl , FormGroup or FormArray . When we use it to update the FormGroup or [FormArray](#) the SetValue requires that the object must match the structure of the FormGroup or FormArray exactly. Otherwise, it will result in an error.

## PatchValue

```
patchValue(value: { [key: string]: any; }, options: { onlySelf?: boolean; emitEvent?: boolean; } = {}): void
```

The PatchValue is used to update only a subset of the elements of the FormGroup or FormArray . It will only update the matching objects and ignores the rest.

## onlySelf

The Angular checks the validation status of the form, whenever there is a change in value. [The validation](#) starts from the control whose value was changed and propagates to the top level FormGroup . This is the default behavior

There may be circumstances, where you do not want angular to check the validity of the entire form, whenever you change the value using the setValue or patchValue . We do that by setting the onlySelf=true as the argument. In such cases, the angular only

checks the validity of the current control, but does not check any other control and does not propagate the validity checking to the parent form group.

## emitEvent

The Angular forms emit two events. One is ValueChanges & the other one is StatusChanges. The ValueChanges event is emitted whenever the value of the form is changed. The StatusChanges event is emitted whenever angular calculates the validation status of the Form. This is the default behavior

We can stop that from happening, by setting the `emitEvent=false`

## SetValue Vs PatchValue

The difference is that with `setValue` we must include all the controls, while with the `patchValue` you can exclude some controls.

## Example form setup

Create a new angular application. Import both `FormsModule`, `ReactiveFormsModule` from `@angular/forms`. Also add it into the imports metadata

```
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
26
```

Create two new components reactive.component.ts & template-component.ts with their respective templates. Also, update the app.component.ts & its template as shown below

[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
```

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

```
1
2 <h1>Angular Forms SetValue & PatchValue Example</h1>
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
```

[tabbyending]

## SetValue & PatchValue in Reactive Forms

Here is our template-component.ts & template-component.html.

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

```
1
2 import { Component, OnInit } from '@angular/core';
3 import { FormGroup, FormControl, Validators } from '@angular/forms'
4
5
6 @Component({
7   templateUrl: './reactive.component.html',
8 })
9 export class ReactiveComponent implements OnInit {
10   title = 'Reactive Forms';
11
12
13   countryList: country[] = [
14     new country("1", "India"),
15     new country('2', 'USA'),
```

```
16   new country('3', 'England')
17 ];
18
19 // reactiveForm = new FormGroup({
20 //   firstname: new FormControl('Sachin'),
21 //   lastname: new FormControl('Tendulkar'),
22 //   email: new FormControl('sachin@gmail.com'),
23 //   gender: new FormControl('male'),
24 //   isMarried: new FormControl(true),
25 //   country: new FormControl('2'),
26 //   address: new FormGroup({
27 //     city: new FormControl("Mumbai"),
28 //     street: new FormControl("Perry Cross Rd"),
29 //     pincode: new FormControl("400050")
30 //   })
31 // })
32
33 reactiveForm = new FormGroup({
34   firstname: new FormControl(),
35   lastname: new FormControl(),
36   email: new FormControl(),
37   gender: new FormControl(),
38   isMarried: new FormControl(),
39   country: new FormControl(),
40   address: new FormGroup({
41     city: new FormControl(),
42     street: new FormControl(),
43     pincode: new FormControl()
44   })
45 })
46
47 onSubmit() {
48   console.log(this.reactiveForm.value);
49 }
50
51 ngOnInit() {
52   this.setDefault();
53 }
54
55 setDefault() {
56
57   let contact = {
58     firstname: "Sachin",
59     lastname: "Tendulkar",
60     email: "sachin@gmail.com",
61     gender: "male",
62     isMarried: true,
63     country: "2",
64     address: {
```

```
65     city: "Mumbai",
66     street: "Perry Cross Rd",
67     pincode: "400050"
68   }
69 };
70
71   this.reactiveForm.setValue(contact);
72 }
73
74 setValue() {
75
76   let contact = {
77     firstname: "Rahul",
78     lastname: "Dravid",
79     email: "rahul@gmail.com",
80     gender: "male",
81     isMarried: true,
82     country: "1",
83     address: {
84       city: "Bangalore",
85       street: "Brigade Road",
86       pincode: "600070"
87     }
88   };
89
90   this.reactiveForm.setValue(contact);
91 }
92
93 setAddress() {
94
95   let address= {
96     city: "Bangalore",
97     street: "Brigade Road",
98     pincode: "600070",
99   };
100
101   this.reactiveForm.get("address").setValue(address);
102
103 };
104
105 setCountry() {
106
107   this.reactiveForm.get("country").setValue("1");
108
109 };
110
111
112 patchAddress() {
113
```



```
114     let address= {
115         city: "Bangalore",
116         street: "Brigade Road",
117         //pincode: "600070",
118         //firstname:'saurv'
119     };
120
121     this.reactiveForm.get("address").patchValue(address);
122
123 }
124
125 patchName() {
126     let contact = {
127         firstname: "Rahul",
128         lastname: "Dravid",
129     }
130
131     this.reactiveForm.patchValue(contact);
132
133 }
134
135 reset() {
136     this.reactiveForm.reset();
137 }
138
139 }
140
141 export class country {
142     id: string;
143     name: string;
144
145     constructor(id: string, name: string) {
146         this.id = id;
147         this.name = name;
148     }
149 }
150
```

[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 </p>
12
13 <p>
14 <label for="lastname">Last Name </label>
15 <input type="text" id="lastname" name="lastname" formControlName="lastname">
16 </p>
17
18 <p>
19 <label for="email">Email </label>
20 <input type="text" id="email" name="email" formControlName="email">
21 </p>
22
23 <p>
24 <label for="gender">Geneder </label>
25 <input type="radio" value="male" id="gender" name="gender" formControlName="g
26 <input type="radio" value="female" id="gender" name="gender" formControlName=
27 </p>
28
29 <p>
30 <label for="isMarried">Married </label>
31 <input type="checkbox" id="isMarried" name="isMarried" formControlName="isMarri
32 </p>
33
34 <p>
35 <label for="country">country </label>
36 <select id="country" name="country" formControlName="country">
37 <option [ngValue]="c.id" *ngFor="let c of countryList">
38 {{c.name}}
39 </option>
40 </select>
41 </p>
42
```

```
43
44 <div formGroupName="address">
45
46   <p>
47     <label for="city">City</label>
48     <input type="text" class="form-control" name="city" formControlName="city">
49   </p>
50
51   <p>
52     <label for="street">Street</label>
53     <input type="text" class="form-control" name="street" formControlName="street">
54   </p>
55
56   <p>
57     <label for="pincode">Pin Code</label>
58     <input type="text" class="form-control" name="pincode" formControlName="pinco
59   </p>
60
61 </div>
62
63
64 <button>Submit</button>
65 <div>
66   <button type="button" (click)="setDefault()">Default</button>
67 </div>
68 <div>
69   <button type="button" (click)="setValue()">SetValue</button>
70   <button type="button" (click)="setAddress()">Address</button>
71   <button type="button" (click)="setCountry()">Country</button>
72 </div>
73 <div>
74   <button type="button" (click)="patchName()">Name</button>
75   <button type="button" (click)="patchAddress()">Address</button>
76   <button type="button" (click)="reset()">Reset</button>
77 </div>
78
79
80 </form>
81 </div>
82
83 <div style="float: right; width:50%;">
84
85 <h3>Form Status</h3>
86 <b>valid : </b>{{reactiveForm.valid}}
87 <b>invalid : </b>{{reactiveForm.invalid}}
88 <b>touched : </b>{{reactiveForm.touched}}
89 <b>untouched : </b>{{reactiveForm.untouched}}
90 <b>pristine : </b>{{reactiveForm.pristine}}
91 <b>dirty : </b>{{reactiveForm.dirty}}
```

```
92 <b>disabled : </b>{{reactiveForm.disabled}}
93 <b>enabled : </b>{{reactiveForm.enabled}}
94
95
96 <h3>Form Value</h3>
97 {{reactiveForm.value |json}}
98
99 </div>
100
```

[tabbyending]

## Setting Initial /Default Value

There are two ways, in which set the initial value. One at the time of defining the Form Model as the first argument to the FormControl as shown below

```
1
2 reactiveForm = new FormGroup({
3   firstname: new FormControl('Sachin'),
4   lastname: new FormControl('Tendulkar'),
5   email: new FormControl('sachin@gmail.com'),
6   gender: new FormControl('male'),
7   isMarried: new FormControl(true),
8   country: new FormControl('2'),
9   address: new FormGroup({
10    city: new FormControl('Mumbai'),
11    street: new FormControl('Perry Cross Rd'),
12    pincode: new FormControl('400050')
13  })
14 })
15
```

Another option is to use the setValue in ngOnInit method. To do that, first, create a contact object with the properties exactly matching the Form Model and then invoke the setValue as shown below

```
1
2 ngOnInit() {
```

```
3  this.setDefault();
4  }
5
6  setDefault() {
7
8      let contact = {
9          firstname: "Sachin",
10         lastname: "Tendulkar",
11         email: "sachin@gmail.com",
12         gender: "male",
13         isMarried: true,
14         country: "2",
15         address: {
16             city: "Mumbai",
17             street: "Perry Cross Rd",
18             pincode: "400050"
19         }
20     };
21
22     this.reactiveForm.setValue(contact);
23 }
24
```

The advantageous of the second option is that you can call the `setDefault` any time and set the default values again.

As said earlier, the `setValue` only works, when the properties match exactly. If you remove any of the properties or add a new property, then it will result in an error.

Ex: if you comment out `isMarried` field, then you will see the following error in the console window.

```
1
2  Must supply a value for form control with name: 'isMarried'.
3
```

Or if you add a new property `surname`, you will see the following error.

```
1  
2 Cannot find form control with name: surname.  
3
```

## Nested FormGroup

As mentioned earlier, the `setValue` updates the entire `FormGroup`. Hence we can update the nested form group separately.

In the following example, we get the reference to the `address` form group and then invoke the `setValue` to update only the address.

```
1  
2 setAddress() {  
3  
4   let address= {  
5     city: "Bangalore",  
6     street: "Brigade Road",  
7     pincode: "600070",  
8   };  
9  
10  this.reactiveForm.get("address").setValue(address);  
11  
12  };  
13
```

Here again, the properties of the address must match completely. Otherwise, it will result in an error.

## FormControl

The value of individual control can be easily set

```
1
2 setCountry() {
3   this.reactiveForm.get("country").setValue("1");
4 };
5
```

## PatchValue

We use `patchValue` when we want to update only the subset of properties.

For Example, the following shows how to update only city & street properties using the `patchValue` method.

```
1
2 patchAddress() {
3
4   let address= {
5     city: "Bangalore",
6     street: "Brigade Road",
7   };
8
9   this.reactiveForm.get("address").patchValue(address);
10 }
11
```

```
1
2
3 patchName() {
4   let contact = {
5     firstname: "Rahul",
6     lastname: "Dravid",
7   }
8
9   this.reactiveForm.patchValue(contact);
10
11 }
```

## Reset Form

```
1  
2 reset() {  
3   this.reactiveForm.reset();  
4 }  
5
```

## OnlySelf Example

The angular forms calculate the validity status of the form, whenever the values of any of the controls on the form change. The validation check starts from the control and is run for the parent control until it reaches the top-level FormGroup.

We can use the `onlySelf:true` argument to tell angular not to run validation on the parent control.

For Example, we have added a required validator to the `firstname` FormControl. Now enter some text in the `firstname` field to make the form Valid and then set the value to blank as shown below. **The Form becomes invalid.**

```
1  
2 withoutOnlySelf(){  
3   this.reactiveForm.get("firstname").setValue("");  
4 }  
5
```

Make the form valid again by entering some text in the `firstname` field. Now, try the same with `onlySelf:true` added. **The Form stays Valid.**

```
1  
2 withOnlySelf(){
```



```
3   this.reactiveForm.get("firstname").setValue("",{onlySelf:true});  
4   }  
5
```

## emitEvent example

The Angular forms emit two events. One is ValueChanges & the Other one is statusChanges . You can stop them from happening using the emitEvent:false argument as shown below.

First, subscribe to statusChanges & valueChanges event at Form Level and also at the control level.

```
1  
2  ngOnInit() {  
3    this.setDefault();  
4  
5    this.reactiveForm.get("firstname").statusChanges.subscribe(x => {  
6      console.log('firstname status changes')  
7    })  
8  
9    this.reactiveForm.get("firstname").valueChanges.subscribe(x => {  
10     console.log('firstname value changed')  
11   })  
12  
13   this.reactiveForm.statusChanges.subscribe(x => {  
14     console.log('form status changes')  
15   })  
16
```

```
17   this.reactiveForm.valueChanges.subscribe(x => {  
18     console.log('form value changed')  
19   })  
20 }  
21
```

And then change the value of the firstname and you will see all the four events are fired.

```
1  
2   withoutEmitEvent(){  
3     this.reactiveForm.get("firstname").setValue("Sachin");  
4   }  
5
```

And when you use the `emitEvent:false` the events are suppressed.

```
1  
2   withEmitEvent(){  
3     this.reactiveForm.get("firstname").setValue("", {emitEvent: false});  
4   }  
5
```

## SetValue & PatchValue in Template-driven Forms

You can make use of the `setValue` & `patchValue` in [template-driven forms](#) also. We learned how to do it in [set Value in template-driven forms in the angular](#) tutorial.

To do that, we first need the reference to the Form model in the template, using the `ViewChild`

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

Once, we have the reference, you can make use of SetValue & PatchValue as shown in the following examples. For a more detailed explanation refer to the tutorial [Set Value in template-driven forms in the angular](#)

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

```
1
2 import { Component, ViewChild, ElementRef, OnInit } 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   countryList: country[] = [
16     new country("1", "India"),
17     new country('2', 'USA'),
18     new country('3', 'England')
19   ];
20
21   contact: contact;
22
23   onSubmit() {
24     console.log(this.templateForm.value);
25   }
26
27   ngOnInit() {
28
29     setTimeout(() => {
30       this.setDefault();
31     });
32   }
33
34   setDefault() {
35
36     let contact = {
37       firstname: "Sachin",
```

```
39     lastname: "Tendulkar",
40     email: "sachin@gmail.com",
41     gender: "male",
42     isMarried: true,
43     country: "2",
44     address: {
45       city: "Mumbai",
46       street: "Perry Cross Rd",
47       pincode: "400050"
48     }
49   };
50
51   this.templateForm.control.setValue(contact);
52 }
53
54
55 setValue() {
56
57   let contact = {
58     firstname: "Rahul",
59     lastname: "Dravid",
60     email: "rahul@gmail.com",
61     gender: "male",
62     isMarried: true,
63     country: "1",
64     address: {
65       city: "Bangalore",
66       street: "Brigade Road",
67       pincode: "600070"
68     }
69   };
70
71   this.templateForm.setValue(contact);
72 }
73
74 setAddress() {
75
76   let address= {
77     city: "Bangalore",
78     street: "Brigade Road",
79     pincode: "600070"
80   };
81
82   this.templateForm.control.get("address").setValue(address);
83
84 };
85
86 setCountry() {
87
```

```
88     let address= {
89         city: "Bangalore",
90         street: "Brigade Road",
91         pincode: "600070"
92     };
93
94     this.templateForm.control.get("country").setValue("1");
95
96 };
97
98
99 patchAddress() {
100
101     let address= {
102         city: "Bangalore",
103         street: "Brigade Road",
104         //pincode: "600070",
105         //firstname:'saurv'
106     };
107
108     this.templateForm.control.get("address").patchValue(address);
109
110 }
111
112 patchName() {
113     let contact = {
114         firstname: "Rahul",
115         lastname: "Dravid",
116     }
117
118     this.templateForm.control.patchValue(contact);
119
120 }
121
122 reset() {
123     this.templateForm.reset();
124 }
125
126 }
127
128 export class contact {
129     firstname:string;
130     lastname:string;
131     gender:string;
132     email:string;
133     isMarried:boolean;
134     country:string;
135     address: {
136         city:string;
```

```
137     street:string;  
138     pincode:string;  
139 }  
140 }  
141  
142  
143 export class country {  
144     id: string;  
145     name: string;  
146  
147     constructor(id: string, name: string) {  
148         this.id = id;  
149         this.name = name;  
150     }  
151 }  
152  
153
```

[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         <p>  
18             <label for="email">Email </label>  
19             <input type="text" id="email" name="email" ngModel>  
20         </p>  
21  
22  
23         <p>  
24             <label for="gender">Geneder </label>  
25             <input type="radio" value="male" id="gender" name="gender" ngModel> Male  
26             <input type="radio" value="female" id="gender" name="gender" ngModel> Female
```

```
27 </p>
28
29 <p>
30   <label for="isMarried">Married </label>
31   <input type="checkbox" id="isMarried" name="isMarried" ngModel>
32 </p>
33
34 <p>
35   <label for="country">country </label>
36   <select id="country" name="country" ngModel>
37     <option [ngValue]="c.id" *ngFor="let c of countryList">
38       {{c.name}}
39     </option>
40   </select>
41 </p>
42
43 <div ngModelGroup="address">
44
45   <p>
46     <label for="city">City</label>
47     <input type="text" id="city" name="city" ngModel>
48   </p>
49
50   <p>
51     <label for="street">Street</label>
52     <input type="text" id="street" name="street" ngModel>
53   </p>
54   <p>
55     <label for="pincode">Pin Code</label>
56     <input type="text" id="pincode" name="pincode" ngModel>
57   </p>
58
59 </div>
60
61 <p>
62   <button type="submit">Submit</button>
63 </p>
64
65 <div>
66   <button type="button" (click)="setDefault()">Default</button>
67 </div>
68 <div>
69   <button type="button" (click)="setValue()">Set Value</button>
70   <button type="button" (click)="setAddress()">Address</button>
71   <button type="button" (click)="setCountry()">Country</button>
72 </div>
73 <div>
74   <button type="button" (click)="patchName()">Name</button>
75   <button type="button" (click)="patchAddress()">Address</button>
```

```
76     <button type="button" (click)="reset()">Reset</button>
77   </div>
78
79
80 </form>
81 </div>
82
83 <div style="float: right; width:50%;">
84   <h3>Form Status</h3>
85   <b>valid : </b>{{templateForm.valid}}
86   <b>invalid : </b>{{templateForm.invalid}}
87   <b>touched : </b>{{templateForm.touched}}
88   <b>untouched : </b>{{templateForm.untouched}}
89   <b>pristine : </b>{{templateForm.pristine}}
90   <b>dirty : </b>{{templateForm.dirty}}
91   <b>disabled : </b>{{templateForm.disabled}}
92   <b>enabled : </b>{{templateForm.enabled}}
93
94
95   <h3>Form Value</h3>
96   {{templateForm.value | json }}
97
98 </div>>
99
```

[tabbyending]

## Summary

In this tutorial, we learned how to use setValue & patchValue to set the values of the [Reactive forms in Angular](#).

Here is the list of all tutorials in Angular Forms

1. [Angular Forms Tutorial: Fundamental & Concepts](#)
2. [Template Driven Forms in Angular](#)
3. [Set Value in Template Driven forms in Angular](#)
4. [Reactive Forms in Angular](#)
5. [FormBuilder in Reactive Forms](#)