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

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

```

```

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { TemplateComponent } from './template-component';
import { ReactiveComponent } from './reactive.component';

@NgModule({
  declarations: [
    AppComponent,TemplateComponent,ReactiveComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }


```

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”]

```

import { Component} from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

[tabby title=”app.component.html”]

```html

<h1>Angular Forms SetValue & PatchValue Example</h1>

<ul>
  <li>
    <a [routerLink]="['/template']" routerLinkActive="router-link-active" >Template</a>
  </li>
  <li>
    <a [routerLink]="['/reactive']" routerLinkActive="router-link-active" >Reactive</a>
  </li>
</ul>

<router-outlet></router-outlet>

```

[tabbyending]

SetValue & PatchValue in Reactive Forms

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

[tabby title=”reactive-component.ts”]

```typescript

import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms'


@Component({
  templateUrl: './reactive.component.html',
})
export class ReactiveComponent implements OnInit {
  title = 'Reactive Forms';


  countryList: country[] = [
    new country("1", "India"),
    new country('2', 'USA'),
    new country('3', 'England')
  ];

  // reactiveForm = new FormGroup({
  //   firstname: new FormControl('Sachin'),
  //   lastname: new FormControl('Tendulkar'),
  //   email: new FormControl('sachin@gmail.com'),
  //   gender: new FormControl('male'),
  //   isMarried: new FormControl(true),
  //   country: new FormControl('2'),
  //   address:new FormGroup({
  //     city: new FormControl("Mumbai"),
  //     street: new FormControl("Perry Cross Rd"),
  //     pincode:new FormControl("400050")
  //   })
  // })

  reactiveForm = new FormGroup({
    firstname: new FormControl(),
    lastname: new FormControl(),
    email: new FormControl(),
    gender: new FormControl(),
    isMarried: new FormControl(),
    country: new FormControl(),
    address:new FormGroup({
      city: new FormControl(),
      street: new FormControl(),
      pincode:new FormControl()
    })
  })

  onSubmit() {
    console.log(this.reactiveForm.value);
  }

  ngOnInit() {
    this.setDefault();
  }

  setDefault() {

    let contact = {
      firstname: "Sachin",
      lastname: "Tendulkar",
      email: "sachin@gmail.com",
      gender: "male",
      isMarried: true,
      country: "2",
      address: {
        city: "Mumbai",
        street: "Perry Cross Rd",
        pincode: "400050"
      }
    };

    this.reactiveForm.setValue(contact);
  }

  setValue() {

    let contact = {
      firstname: "Rahul",
      lastname: "Dravid",
      email: "rahul@gmail.com",
      gender: "male",
      isMarried: true,
      country: "1",
      address: {
        city: "Bangalore",
        street: "Brigade Road",
        pincode: "600070"
      }
    };

    this.reactiveForm.setValue(contact);
  }

  setAddress() {

    let address= {
      city: "Bangalore",
      street: "Brigade Road",
      pincode: "600070",
    };

    this.reactiveForm.get("address").setValue(address);

  };

  setCountry() {

    this.reactiveForm.get("country").setValue("1");

  };


  patchAddress() {

    let address= {
      city: "Bangalore",
      street: "Brigade Road",
      //pincode: "600070",
      //firstname:'saurv'
    };

    this.reactiveForm.get("address").patchValue(address);

  }

  patchName() {
    let contact = {
      firstname: "Rahul",
      lastname: "Dravid",
    }

    this.reactiveForm.patchValue(contact);

  }

  reset() {
    this.reactiveForm.reset();
  }

}

export class country {
  id: string;
  name: string;

  constructor(id: string, name: string) {
    this.id = id;
    this.name = name;
  }
}

```

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

```

<h3>{{title}}</h3>

<div style="float: left; width:50%;">

  <form [formGroup]="reactiveForm" (ngSubmit)="onSubmit()" novalidate>

    <p>
      <label for="firstname">First Name </label>
      <input type="text" id="firstname" name="firstname" formControlName="firstname">
    </p>

    <p>
      <label for="lastname">Last Name </label>
      <input type="text" id="lastname" name="lastname" formControlName="lastname">
    </p>

    <p>
      <label for="email">Email </label>
      <input type="text" id="email" name="email" formControlName="email">
    </p>

    <p>
      <label for="gender">Geneder </label>
      <input type="radio" value="male" id="gender" name="gender" formControlName="g
      <input type="radio" value="female" id="gender" name="gender" formControlName=
    </p>

    <p>
      <label for="isMarried">Married </label>
      <input type="checkbox" id="isMarried" name="isMarried" formControlName="isMarri
    </p>

    <p>
      <label for="country">country </label>
      <select id="country" name="country" formControlName="country">
        <option [ngValue]="c.id" *ngFor="let c of countryList">
          {{c.name}}
        </option>
      </select>
    </p>


    <div formGroupName="address">

      <p>
        <label for="city">City</label>
        <input type="text" class="form-control" name="city" formControlName="city">
      </p>

      <p>
        <label for="street">Street</label>
        <input type="text" class="form-control" name="street" formControlName="street"
      </p>

      <p>
        <label for="pincode">Pin Code</label>
        <input type="text" class="form-control" name="pincode" formControlName="pinco
      </p>

    </div>


    <button>Submit</button>
    <div>
      <button type="button" (click)="setDefault()">Default</button>
    </div>
    <div>
      <button type="button" (click)="setValue()">SetValue</button>
      <button type="button" (click)="setAddress()">Address</button>
      <button type="button" (click)="setCountry()">Country</button>
    </div>
    <div>
      <button type="button" (click)="patchName()">Name</button>
      <button type="button" (click)="patchAddress()">Address</button>
      <button type="button" (click)="reset()">Reset</button>
    </div>


  </form>
</div>

<div style="float: right; width:50%;">

  <h3>Form Status</h3>
  <b>valid : </b>{{reactiveForm.valid}}
  <b>invalid : </b>{{reactiveForm.invalid}}
  <b>touched : </b>{{reactiveForm.touched}}
  <b>untouched : </b>{{reactiveForm.untouched}}
  <b>pristine : </b>{{reactiveForm.pristine}}
  <b>dirty : </b>{{reactiveForm.dirty}}
  <b>disabled : </b>{{reactiveForm.disabled}}
  <b>enabled : </b>{{reactiveForm.enabled}}


  <h3>Form Value</h3>
  {{reactiveForm.value |json}}

</div>

```

[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

```

reactiveForm = new FormGroup({
    firstname: new FormControl('Sachin'),
    lastname: new FormControl('Tendulkar'),
    email: new FormControl('sachin@gmail.com'),
    gender: new FormControl('male'),
    isMarried: new FormControl(true),
    country: new FormControl('2'),
    address:new FormGroup({
      city: new FormControl("Mumbai"),
      street: new FormControl("Perry Cross Rd"),
      pincode:new FormControl("400050")
    })
  })

```

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

```

ngOnInit() {
  this.setDefault();
}

setDefault() {

  let contact = {
    firstname: "Sachin",
    lastname: "Tendulkar",
    email: "sachin@gmail.com",
    gender: "male",
    isMarried: true,
    country: "2",
    address: {
      city: "Mumbai",
      street: "Perry Cross Rd",
      pincode: "400050"
    }
  };

  this.reactiveForm.setValue(contact);
 }

```

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.

```

Must supply a value for form control with name: 'isMarried'.

```

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

```

Cannot find form control with name: surname.

```

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.

```

  setAddress() {

    let address= {
      city: "Bangalore",
      street: "Brigade Road",
      pincode: "600070",
    };

    this.reactiveForm.get("address").setValue(address);

  };

```

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

```

setCountry() {
    this.reactiveForm.get("country").setValue("1");
};

```

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.

```

 patchAddress() {

   let address= {
     city: "Bangalore",
     street: "Brigade Road",
   };

   this.reactiveForm.get("address").patchValue(address);
 }

```

```


  patchName() {
    let contact = {
      firstname: "Rahul",
      lastname: "Dravid",
    }

    this.reactiveForm.patchValue(contact);

  }
```

```

```

Reset Form

```

reset() {
  this.reactiveForm.reset();
}

```

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**.

```

  withOutOnlySelf(){
    this.reactiveForm.get("firstname").setValue("");
  }

```

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**.

```

  withOnlySelf(){
```

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

```

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.

```

ngOnInit() {
  this.setDefault();

  this.reactiveForm.get("firstname").statusChanges.subscribe(x => {
    console.log('firstname status changes')
  })

  this.reactiveForm.get("firstname").valueChanges.subscribe(x => {
    console.log('firstname value changed')
  })

  this.reactiveForm.statusChanges.subscribe(x => {
    console.log('form status changes')
  })

```

```
    this.reactiveForm.valueChanges.subscribe(x => {
      console.log('form value changed')
    })
  }

```

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

```

withouEmitEvent(){
  this.reactiveForm.get("firstname").setValue("Sachin");
}

```

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

```

withEmitEvent(){
  this.reactiveForm.get("firstname").setValue("",{emitEvent:false});
}

```

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

```

 @ViewChild('templateForm',null) templateForm: NgForm;

```

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"]

```
import { Component, ViewChild, ElementRef, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';


@Component({
  templateUrl: './template.component.html',
})
export class TemplateComponent implements OnInit {

  title = 'Template driven forms';

  @ViewChild('templateForm',null) templateForm: NgForm;

  countryList: country[] = [
    new country("1", "India"),
    new country('2', 'USA'),
    new country('3', 'England')
  ];

  contact: contact;

  onSubmit() {
    console.log(this.templateForm.value);
  }

  ngOnInit() {

    setTimeout(() => {
      this.setDefault();
    });

  }

  setDefault() {

    let contact = {
      firstname: "Sachin",
      lastname: "Tendulkar",
      email: "sachin@gmail.com",
      gender: "male",
      isMarried: true,
      country: "2",
      address: {
        city: "Mumbai",
        street: "Perry Cross Rd",
        pincode: "400050"
      }
    };

    this.templateForm.control.setValue(contact);
  }


  setValue() {

    let contact = {
      firstname: "Rahul",
      lastname: "Dravid",
      email: "rahul@gmail.com",
      gender: "male",
      isMarried: true,
      country: "1",
      address: {
        city: "Bangalore",
        street: "Brigade Road",
        pincode: "600070"
      }
    };

    this.templateForm.setValue(contact);
  }

  setAddress() {

    let address= {
      city: "Bangalore",
      street: "Brigade Road",
      pincode: "600070"
    };

    this.templateForm.control.get("address").setValue(address);

  };

  setCountry() {

    let address= {
      city: "Bangalore",
      street: "Brigade Road",
      pincode: "600070"
    };

    this.templateForm.control.get("country").setValue("1");

  };


  patchAddress() {

    let address= {
      city: "Bangalore",
      street: "Brigade Road",
      //pincode: "600070",
      //firstname:'saurv'
    };

    this.templateForm.control.get("address").patchValue(address);

  }

  patchName() {
    let contact = {
      firstname: "Rahul",
      lastname: "Dravid",
    }

    this.templateForm.control.patchValue(contact);

  }

  reset() {
    this.templateForm.reset();
  }

}

export class contact {
  firstname:string;
  lastname:string;
  gender:string;
  email:string;
  isMarried:boolean;
  country:string;
  address: {
    city:string;
    street:string;
    pincode:string;
  }
}


export class country {
  id: string;
  name: string;

  constructor(id: string, name: string) {
    this.id = id;
    this.name = name;
  }
}


```

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

```html

<h3>{{title}}</h3>

<div style="float: left; width:50%;">
  <form #templateForm="ngForm" (ngSubmit)="onSubmit(templateForm)">

    <p>
      <label for="firstname">First Name </label>
      <input type="text" id="firstname" name="firstname" #fname="ngModel" ngModel>

    </p>
    <p>
      <label for="lastname">Last Name </label>
      <input type="text" id="lastname" name="lastname" ngModel>
    </p>

    <p>
      <label for="email">Email </label>
      <input type="text" id="email" name="email" ngModel>
    </p>


    <p>
      <label for="gender">Geneder </label>
      <input type="radio" value="male" id="gender" name="gender" ngModel> Male
      <input type="radio" value="female" id="gender" name="gender" ngModel> Female
  </p>

  <p>
    <label for="isMarried">Married </label>
    <input type="checkbox" id="isMarried" name="isMarried" ngModel>
  </p>

  <p>
    <label for="country">country </label>
    <select id="country" name="country" ngModel>
      <option [ngValue]="c.id" *ngFor="let c of countryList">
        {{c.name}}
      </option>
    </select>
  </p>

  <div ngModelGroup="address">

    <p>
      <label for="city">City</label>
      <input type="text" id="city" name="city" ngModel>
    </p>

    <p>
      <label for="street">Street</label>
      <input type="text" id="street" name="street" ngModel>
    </p>
    <p>
      <label for="pincode">Pin Code</label>
      <input type="text" id="pincode" name="pincode" ngModel>
    </p>

  </div>

  <p>
    <button type="submit">Submit</button>
  </p>

  <div>
    <button type="button" (click)="setDefault()">Default</button>
  </div>
  <div>
    <button type="button" (click)="setValue()">SetValue</button>
    <button type="button" (click)="setAddress()">Address</button>
    <button type="button" (click)="setCountry()">Country</button>
  </div>
  <div>
    <button type="button" (click)="patchName()">Name</button>
    <button type="button" (click)="patchAddress()">Address</button>
```

```
      <button type="button" (click)="reset()">Reset</button>
    </div>


  </form>
</div>

<div style="float: right; width:50%;">
  <h3>Form Status</h3>
  <b>valid : </b>{{templateForm.valid}}
  <b>invalid : </b>{{templateForm.invalid}}
  <b>touched : </b>{{templateForm.touched}}
  <b>untouched : </b>{{templateForm.untouched}}
  <b>pristine : </b>{{templateForm.pristine}}
  <b>dirty : </b>{{templateForm.dirty}}
  <b>disabled : </b>{{templateForm.disabled}}
  <b>enabled : </b>{{templateForm.enabled}}


  <h3>Form Value</h3>
  {{templateForm.value | json }}

</div>>

```

[tabbyending]

Summary

In this tutorial, we learned how to use setValue & pathcValue 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