Using Angular FormBuilder to build Forms

**Table of Contents**

* [What is FormBuilder](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#what-is-formbuilder)
* [How to use FormBuilder](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#how-to-use-formbuilder)
  + [Import & inject FormBuilder API](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#import-inject-formbuilder-api)
  + [FormGroup](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#formgroup)
  + [Nested FormGroup](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#nested-formgroup)
  + [Validations](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#validations)
* [FormBuilder Example](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#formbuilder-example)
* [References](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#references)
* [Summary](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/#summary)

## What is FormBuilder

The FormBuilder is the helper API to build forms in Angular.  It provides shortcuts to create the instance of the FormControl, [FormGroup](https://www.tektutorialshub.com/angular/formgroup-in-angular/) or FormArray. It reduces the code required to write the complex forms.

## How to use FormBuilder

### Import & inject FormBuilder API

To use the FormBuilder, first, we need to import it in our component

|  |  |
| --- | --- |
| 1  2  3 | import { FormBuilder } from '@angular/forms' |

Next, we need to inject it into our component class

|  |  |
| --- | --- |
| 1  2  3  4 | constructor(private formBuilder: FormBuilder) {  } |

Finally, use the group, array & control methods to build the FormModel

### FormGroup

We use the group method to build the Form Group. We pass the list of [Form Controls](https://www.tektutorialshub.com/angular/formcontrol-in-angular/), [Form Array](https://www.tektutorialshub.com/angular/angular-formarray-example-in-reactive-forms/), or another [Form Group](https://www.tektutorialshub.com/angular/formgroup-in-angular/) to the group method as key-value pair. Where the key is the name of the FormControl, FormGroup or FormArray. The value is the configuration of the control.

In the following example, we have added six form controls. The First argument to the FormControl is the initial value, which we set to empty string.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | this.contactForm = this.formBuilder.group({    firstname: [''],    lastname: [''],    email: [''],    gender: [''],    isMarried: [''],    country: [''],  }); |

### Nested FormGroup

Creating a Nested FormGroup is just as easy. use the formbuilder.group method, as shown below.

this.contactForm = this.formBuilder.group({

  firstname: [''],

  lastname: [''],

  email: [''],

  gender: [''],

  isMarried: [''],

  country: [''],

  address: this.formBuilder.group({

    city: [''],

    street: [''],

    pincode: ['']

  })

})

### Validations

The second argument to the FormControl is the list of sync validators. The following example shows how to add validators.

this.contactForm = this.formBuilder.group({

  firstname: ['', [Validators.required, Validators.minLength(10)]],

  lastname: ['', [Validators.required, Validators.maxLength(15), Validators.pattern("^[a-zA-Z]+$")]],

  email: ['', [Validators.required, Validators.email]],

  gender: ['', [Validators.required]],

  isMarried: ['', [Validators.required]],

  country: ['', [Validators.required]],

  address: this.formBuilder.group({

    city: ['', [Validators.required]],

    street: ['', [Validators.required]],

    pincode: ['', [Validators.required]],

  })

});

## FormBuilder Example

We learned how to build reactive forms the [Angular Reactive forms tutorial](https://www.tektutorialshub.com/angular/angular-reactive-forms/).

**app.component.ts**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123 | import { Component, ViewChild, ElementRef } from '@angular/core';  import { FormGroup, FormControl, Validators } from '@angular/forms'  import { FormBuilder } from '@angular/forms'  import { groupBy } from 'rxjs/internal/operators/groupBy';      @Component({    selector: 'app-root',    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {      title = 'Angular Reactive forms';      contactForm;      constructor(private formBuilder: FormBuilder) {            // this.contactForm = this.formBuilder.group({      //   firstname: [''],      //   lastname: [''],      //   email: [''],      //   gender: [''],      //   isMarried: [''],      //   country: [''],      // });        // this.contactForm = this.formBuilder.group({      //   firstname: [''],      //   lastname: [''],      //   email: [''],      //   gender: [''],      //   isMarried: [''],      //   country: [''],      //   address: this.formBuilder.group({      //     city: [''],      //     street: [''],      //     pincode: ['']      //   })      // });        this.contactForm = this.formBuilder.group({        firstname: ['', [Validators.required, Validators.minLength(10)]],        lastname: ['', [Validators.required, Validators.maxLength(15), Validators.pattern("^[a-zA-Z]+$")]],        email: ['', [Validators.required, Validators.email]],        gender: ['', [Validators.required]],        isMarried: ['', [Validators.required]],        country: ['', [Validators.required]],        address: this.formBuilder.group({          city: ['', [Validators.required]],          street: ['', [Validators.required]],          pincode: ['', [Validators.required]],        })      });    }        get firstname() {      return this.contactForm.get('firstname');    }      get lastname() {      return this.contactForm.get('lastname');    }      get email() {      return this.contactForm.get('email');    }      get gender() {      return this.contactForm.get('gender');    }      get isMarried() {      return this.contactForm.get('isMarried');    }      get country() {      return this.contactForm.get('country');    }      get city() {      return this.contactForm.get("address").get('city');    }      get street() {      return this.contactForm.get("address").get('street');    }      get pincode() {      return this.contactForm.get("address").get('pincode');    }        countryList: country[] = [      new country("1", "India"),      new country('2', 'USA'),      new country('3', 'England')    ];      onSubmit() {      console.log(this.contactForm.value);    }    }      export class country {    id: string;    name: string;      constructor(id: string, name: string) {      this.id = id;      this.name = name;    }  } |

**app.component.html**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147 | <div style="float: left; width:50%;">    <form [formGroup]="contactForm" (ngSubmit)="onSubmit()" novalidate>      <p>      <label for="firstname">First Name </label>      <input type="text" id="firstname" name="firstname" formControlName="firstname">    </p>      <div      \*ngIf="!firstname?.valid && (firstname?.dirty ||firstname?.touched)">      <div [hidden]="!firstname.errors.required">        First Name is required      </div>      <div [hidden]="!firstname.errors.minlength">        Min Length is 10      </div>    </div>      <p>      <label for="lastname">Last Name </label>      <input type="text" id="lastname" name="lastname" formControlName="lastname">    </p>      <div \*ngIf="!lastname.valid && (lastname.dirty ||lastname.touched)">      <div [hidden]="!lastname.errors.pattern">        Only characters are allowed      </div>      <div [hidden]="!lastname.errors.maxLength">        Max length allowed is {{lastname.errors.maxlength?.requiredLength}}      </div>      <div [hidden]="!lastname.errors.required">        Last Name is required      </div>    </div>      <p>      <label for="email">Email </label>      <input type="text" id="email" name="email" formControlName="email">    </p>    <div \*ngIf="!email.valid && (email.dirty ||email.touched)">      <div [hidden]="!email.errors.required">        email is required      </div>      <div [hidden]="!email.errors.email">        invalid email id      </div>    </div>        <p>      <label for="gender">Geneder </label>      <input type="radio" value="male" id="gender" name="gender" formControlName="gender"> Male      <input type="radio" value="female" id="gender" name="gender" formControlName="gender"> Female    </p>    <div \*ngIf="!gender.valid && (gender.dirty ||gender.touched)">      <div [hidden]="!gender.errors.required">        gender is required      </div>    </div>      <p>      <label for="isMarried">Married </label>      <input type="checkbox" id="isMarried" name="isMarried" formControlName="isMarried">    </p>    <div \*ngIf="!isMarried.valid && (isMarried.dirty ||isMarried.touched)">      <div [hidden]="!isMarried.errors.required">        isMarried is required      </div>    </div>        <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 \*ngIf="!country.valid && (country.dirty ||country.touched)">      <div [hidden]="!country.errors.required">        country is required      </div>    </div>      <div formGroupName="address">        <div class="form-group">        <label for="city">City</label>        <input type="text" class="form-control" name="city" formControlName="city">      </div>      <div \*ngIf="!city.valid && (city.dirty ||city.touched)">        <div [hidden]="!city.errors.required">          city is required        </div>      </div>          <div class="form-group">        <label for="street">Street</label>        <input type="text" class="form-control" name="street" formControlName="street">      </div>      <div \*ngIf="!street.valid && (street.dirty ||street.touched)">        <div [hidden]="!street.errors.required">          street is required        </div>      </div>        <div class="form-group">        <label for="pincode">Pin Code</label>        <input type="text" class="form-control" name="pincode" formControlName="pincode">      </div>      <div \*ngIf="!pincode.valid && (pincode.dirty ||pincode.touched)">        <div [hidden]="!pincode.errors.required">          pincode is required        </div>      </div>      </div>      <p>      <button type="submit" [disabled]="!contactForm.valid">Submit</button>    </p>    </form>    </div>    <div style="float: right; width:50%;">      <h3>Form Status</h3>    <b>valid : </b>{{contactForm.valid}}    <b>invalid : </b>{{contactForm.invalid}}    <b>touched : </b>{{contactForm.touched}}    <b>untouched : </b>{{contactForm.untouched}}    <b>pristine : </b>{{contactForm.pristine}}    <b>dirty : </b>{{contactForm.dirty}}    <b>disabled : </b>{{contactForm.disabled}}    <b>enabled : </b>{{contactForm.enabled}}      <h3>Form Value</h3>    {{contactForm.value |json}}    </div> |

## References

1. [FormBuilder API](https://angular.io/api/forms/FormBuilder)

## Summary

FormBuilder API makes it easier to work with the reactive forms in Angular. We can make use of the group, array & control methods to build ourFormModel. FormBuilder reduces the code required to write the complex forms.

1. [Angular Forms Tutorial: Fundamental & Concepts](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/)
2. [Template Driven Forms in Angular](https://www.tektutorialshub.com/angular/angular-template-driven-forms/)
3. [Set Value in Template Driven forms in Angular](https://www.tektutorialshub.com/angular/how-to-set-value-in-template-driven-forms-in-angular/)
4. [Reactive Forms in Angular](https://www.tektutorialshub.com/angular/angular-reactive-forms/)
5. [FormBuilder in Reactive Forms](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/)
6. [SetValue & PatchValue in Angular](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/)
7. [StatusChanges in Angular Forms](https://www.tektutorialshub.com/angular/statuschanges-in-angular-forms/)
8. [ValueChanges in Angular Forms](https://www.tektutorialshub.com/angular/valuechanges-in-angular-forms/)
9. [FormControl](https://www.tektutorialshub.com/angular/formcontrol-in-angular/)
10. [FormGroup](https://www.tektutorialshub.com/angular/formgroup-in-angular/)
11. [FormArray Example](https://www.tektutorialshub.com/angular/angular-formarray-example-in-reactive-forms/)
12. [Build Dynamic or Nested Forms using FormArray](https://www.tektutorialshub.com/angular/nested-formarray-example-add-form-fields-dynamically/)
13. [Validations in Reactive Forms in Angular](https://www.tektutorialshub.com/angular/angular-reactive-forms-validation/)
14. [Custom Validator in Reactive Forms](https://www.tektutorialshub.com/angular/custom-validator-in-angular-reactive-form/)
15. [Passing Parameter to Custom Validator in Reactive Forms](https://www.tektutorialshub.com/angular/custom-validator-with-parameters-in-angular/)
16. [Inject Service into Custom Validator](https://www.tektutorialshub.com/angular/inject-service-into-validator-in-angular/)
17. [Validation in Template Driven Forms](https://www.tektutorialshub.com/angular/template-driven-form-validation-in-angular/)
18. [Custom Validator in Template Driven Forms](https://www.tektutorialshub.com/angular/custom-validator-in-template-driven-forms-in-angular/)

# SetValue & PatchValue in Angular

In this tutorial, we will learn how to set the model values in [Reactive Forms](https://www.tektutorialshub.com/angular/angular-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](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#angular-forms)
* [SetValue](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#setvalue)
* [PatchValue](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#patchvalue)
  + [onlySelf](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#onlyself)
  + [emitEvent](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#emitevent)
* [SetValue Vs PatchValue](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#setvalue-vs-patchvalue)
* [Example form setup](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#example-form-setup)
* [SetValue & PatchValue in Reactive Forms](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#setvalue-patchvalue-in-reactive-forms)
  + [Setting Initial /Default Value](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#setting-initial-default-value)
  + [Nested FormGroup](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#nested-formgroup)
  + [FormControl](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#formcontrol)
* [PatchValue](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#patchvalue-2)
* [Reset Form](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#reset-form)
* [OnlySelf Example](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#onlyself-example)
* [emitEvent example](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#emitevent-example)
* [SetValue & PatchValue in Template-driven Forms](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#setvalue-patchvalue-in-template-driven-forms)
* [Summary](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/#summary)

## Angular Forms

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

1. [Angular Forms Tutorial: Fundamental & Concepts](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/)
2. [Template Driven Forms in Angular](https://www.tektutorialshub.com/angular/angular-template-driven-forms/)
3. [Set Value in Template Driven forms in Angular](https://www.tektutorialshub.com/angular/how-to-set-value-in-template-driven-forms-in-angular/)
4. [Reactive Forms in Angular](https://www.tektutorialshub.com/angular/angular-reactive-forms/)
5. [FormBuilder in Reactive Forms](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/)
6. [SetValue & PatchValue in Angular](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/)
7. [StatusChanges in Angular Forms](https://www.tektutorialshub.com/angular/statuschanges-in-angular-forms/)
8. [ValueChanges in Angular Forms](https://www.tektutorialshub.com/angular/valuechanges-in-angular-forms/)
9. [FormControl](https://www.tektutorialshub.com/angular/formcontrol-in-angular/)
10. [FormGroup](https://www.tektutorialshub.com/angular/formgroup-in-angular/)
11. [FormArray Example](https://www.tektutorialshub.com/angular/angular-formarray-example-in-reactive-forms/)
12. [Build Dynamic or Nested Forms using FormArray](https://www.tektutorialshub.com/angular/nested-formarray-example-add-form-fields-dynamically/)
13. [Validations in Reactive Forms in Angular](https://www.tektutorialshub.com/angular/angular-reactive-forms-validation/)
14. [Custom Validator in Reactive Forms](https://www.tektutorialshub.com/angular/custom-validator-in-angular-reactive-form/)
15. [Passing Parameter to Custom Validator in Reactive Forms](https://www.tektutorialshub.com/angular/custom-validator-with-parameters-in-angular/)
16. [Inject Service into Custom Validator](https://www.tektutorialshub.com/angular/inject-service-into-validator-in-angular/)
17. [Validation in Template Driven Forms](https://www.tektutorialshub.com/angular/template-driven-form-validation-in-angular/)
18. [Custom Validator in Template Driven Forms](https://www.tektutorialshub.com/angular/custom-validator-in-template-driven-forms-in-angular/)

The [Angular Forms](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/) 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](https://www.tektutorialshub.com/angular/angular-formarray-example-in-reactive-forms/) the SetValue requires that the object must match the structure of the FormGroup or FormArray exactly. Otherwise, it will result in an error.

[**BEST ANGULAR BOOKS**](https://www.tektutorialshub.com/angular/angular-best-books/)  
**The Top 8**[**Best Angular Books**](https://www.tektutorialshub.com/angular/angular-best-books/)**, which helps you to get started with Angular**

## 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](https://www.tektutorialshub.com/angular/formgroup-in-angular/) 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](https://www.tektutorialshub.com/angular/valuechanges-in-angular-forms/) & 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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | 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”]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | 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”]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <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”]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150 | 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”]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100 | <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="gender"> Male        <input type="radio" value="female" id="gender" name="gender" formControlName="gender"> Female      </p>        <p>        <label for="isMarried">Married </label>        <input type="checkbox" id="isMarried" name="isMarried" formControlName="isMarried">      </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="pincode">        </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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | 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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | 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.

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

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

|  |  |
| --- | --- |
| 1  2  3 | 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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | 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

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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.

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | patchName() {      let contact = {        firstname: "Rahul",        lastname: "Dravid",      }        this.reactiveForm.patchValue(contact);      } |

## Reset Form

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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**.

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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**.

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | 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.

|  |  |
| --- | --- |
| 1  2  3  4  5 | withouEmitEvent(){    this.reactiveForm.get("firstname").setValue("Sachin");  } |

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

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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](https://www.tektutorialshub.com/angular/angular-template-driven-forms/) also. We learned how to do it in [set Value in template-driven forms in the angular](https://www.tektutorialshub.com/angular/how-to-set-value-in-template-driven-forms-in-angular/) tutorial.

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

|  |  |
| --- | --- |
| 1  2  3 | @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](https://www.tektutorialshub.com/angular/how-to-set-value-in-template-driven-forms-in-angular/)

[tabby title=”template-component.ts”]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | 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”]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 | <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](https://www.tektutorialshub.com/angular/angular-reactive-forms/).

Here is the list of all tutorials in Angular Forms

1. [Angular Forms Tutorial: Fundamental & Concepts](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/)
2. [Template Driven Forms in Angular](https://www.tektutorialshub.com/angular/angular-template-driven-forms/)
3. [Set Value in Template Driven forms in Angular](https://www.tektutorialshub.com/angular/how-to-set-value-in-template-driven-forms-in-angular/)
4. [Reactive Forms in Angular](https://www.tektutorialshub.com/angular/angular-reactive-forms/)
5. [FormBuilder in Reactive Forms](https://www.tektutorialshub.com/angular/angular-formbuilder-in-reactive-forms/)
6. [SetValue & PatchValue in Angular](https://www.tektutorialshub.com/angular/setvalue-patchvalue-in-angular/)
7. [StatusChanges in Angular Forms](https://www.tektutorialshub.com/angular/statuschanges-in-angular-forms/)
8. [ValueChanges in Angular Forms](https://www.tektutorialshub.com/angular/valuechanges-in-angular-forms/)
9. [FormControl](https://www.tektutorialshub.com/angular/formcontrol-in-angular/)
10. [FormGroup](https://www.tektutorialshub.com/angular/formgroup-in-angular/)
11. [FormArray Example](https://www.tektutorialshub.com/angular/angular-formarray-example-in-reactive-forms/)
12. [Build Dynamic or Nested Forms using FormArray](https://www.tektutorialshub.com/angular/nested-formarray-example-add-form-fields-dynamically/)
13. [Validations in Reactive Forms in Angular](https://www.tektutorialshub.com/angular/angular-reactive-forms-validation/)
14. [Custom Validator in Reactive Forms](https://www.tektutorialshub.com/angular/custom-validator-in-angular-reactive-form/)
15. [Passing Parameter to Custom Validator in Reactive Forms](https://www.tektutorialshub.com/angular/custom-validator-with-parameters-in-angular/)
16. [Inject Service into Custom Validator](https://www.tektutorialshub.com/angular/inject-service-into-validator-in-angular/)
17. [Validation in Template Driven Forms](https://www.tektutorialshub.com/angular/template-driven-form-validation-in-angular/)
18. [Custom Validator in Template Driven Forms](https://www.tektutorialshub.com/angular/custom-validator-in-template-driven-forms-in-angular/)