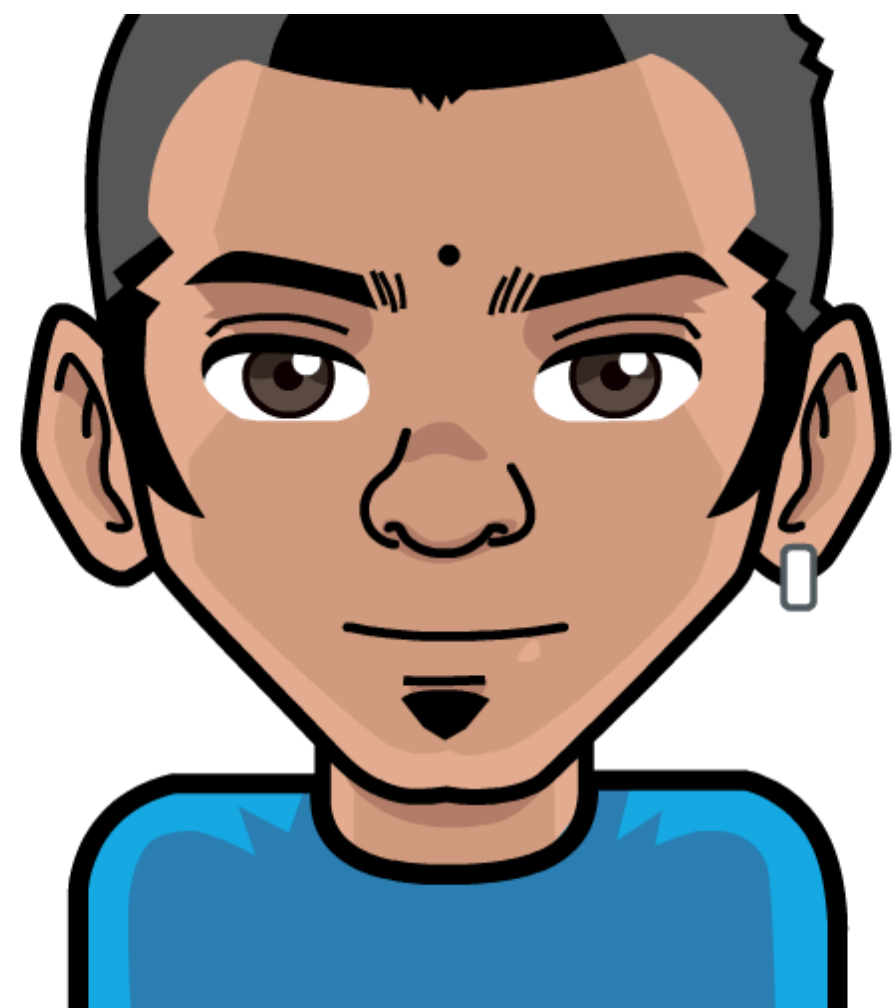





RAJU GANDHI

---

# FORMING ANGULAR FORMS



# RAJU GANDHI

   @LOOSELYTYPED  
CTO - INTEGRALLIS SOFTWARE

# ANGULAR FORMS

**TEMPLATE DRIVEN**

# CHARACTERISTICS

- ▶ Familiar
- ▶ Simple use cases
- ▶ Harder to test

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

```
@NgModule({
  declarations: [
    // ...
  ],
  imports: [
    BrowserModule,
    FormsModule,
  ],
  providers: [
    // ...
  ],
  bootstrap: [
    AppComponent
  ]
})
export class AppModule { }
```

ngForm = <form> + 

```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
        required>
    </div>
    <button type="submit"
      class="btn btn-success"
      [disabled]="!addNewPersonForm.valid">Submit</button>

    {{ addNewPersonForm.value | json }}

  </form>
</div>
```



```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit()"
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
        required>
    </div>
    <button type="submit"
      class="btn btn-success"
      [disabled]="!addNewPersonForm.valid">Submit</button>

    {{ addNewPersonForm.value | json }}

  </form>
</div>
```

template local variable

```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit(addNewPersonForm)">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
        required>
    </div>
    <button type="submit"
      class="btn btn-success"
      [disabled]="!addNewPersonForm.valid">Submit</button>

    <p>value {{ addNewPersonForm.value | json }}</p>
    <p>value {{ addNewPersonForm.value.firstName | json }}</p>
    <p>pristine {{ addNewPersonForm.pristine | json }}</p>
    <p>untouched {{ addNewPersonForm.untouched | json }}</p>
    <p>valid {{ addNewPersonForm.valid | json }}</p>
  </form>
</div>
```

```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit(addNewPersonForm)">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
        required>
    </div>
    <button type="submit"
      class="btn btn-success"
      [disabled]="!addNewPersonForm.valid">Submit</button>

    <p>value {{ addNewPersonForm.value | json}}</p>
    <p>value {{ addNewPersonForm.value.firstName | json}}</p>
    <p>pristine {{ addNewPersonForm.pristine | json}}</p>
    <p>untouched {{ addNewPersonForm.untouched | json}}</p>
    <p>valid {{ addNewPersonForm.valid | json}}</p>
  </form>
</div>
```

Any control fail validation?

valid

invalid

Any control value changed?

pristine

dirty

Any control visited?

untouched

touched

Any control fail validation?

ng-valid

ng-invalid

Any control value changed?

ng-pristine

ng-dirty

Any control visited?

ng-untouched

ng-touched

```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit(addNewPersonForm)">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
        #firstName="ngModel"
        required>
      <div [hidden]="firstName.valid || firstName.pristine"
        class="alert alert-danger">
        First name is required
      </div>
    </div>
  </form>
</div>
```

```
<form #addNewPersonForm="ngForm"
      (ngSubmit)="onSubmit(addNewPersonForm)">
  <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
  </form>
```



```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit(addNewPersonForm)">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        ngModel
        name="firstName"
        #firstName="ngModel"
        required>
      <div [hidden]="firstName.valid || firstName.pristine"
        class="alert alert-danger">
        First name is required
      </div>
    </div>
  </form>
</div>
```



```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit(addNewPersonForm)">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        [(ngModel)]="model.first_name"
        name="firstName"
        #firstName="ngModel"
        required>
      <div [hidden]="firstName.valid || firstName.pristine"
        class="alert alert-danger">
        First name is required
      </div>
    </div>
  </form>
</div>
```

```
<form #addNewPersonForm="ngForm"
      (ngSubmit)="onSubmit(addNewPersonForm)">
  <input type="text"
        class="form-control"
        id="first-name"
        [(ngModel)]="model.first_name"
        name="firstName"
  </form>
```



```
<div class="col-xs-12 col-md-12">
  <h1>Add a new person</h1>
  <form #addNewPersonForm="ngForm"
    (ngSubmit)="onSubmit(addNewPersonForm)">
    <div class="form-group">
      <label for="first-name">First name</label>
      <input type="text"
        class="form-control"
        id="first-name"
        [ngModel]="model.firstName"
        name="firstName"
        #firstName="ngModel"
        required>
      <div [hidden]="firstName.valid || firstName.pristine"
        class="alert alert-danger">
        First name is required
      </div>
    </div>
  </form>
</div>
```

```
<form #addNewPersonForm="ngForm"
      (ngSubmit)="onSubmit(addNewPersonForm)">
  <input type="text"
        class="form-control"
        id="first-name"
        [ngModel]="model.first_name"
        name="firstName"

</form>
```





# NGFORM

- ▶ Optional
- ▶ ngNoForm is also available
- ▶ In-build validators
  - ▶ RequiredValidator
  - ▶ MinLengthValidator
  - ▶ MaxLengthValidator
  - ▶ PatternValidator
  - ▶ CheckboxRequiredValidator
  - ▶ EmailValidator

**MODEL DRIVEN**

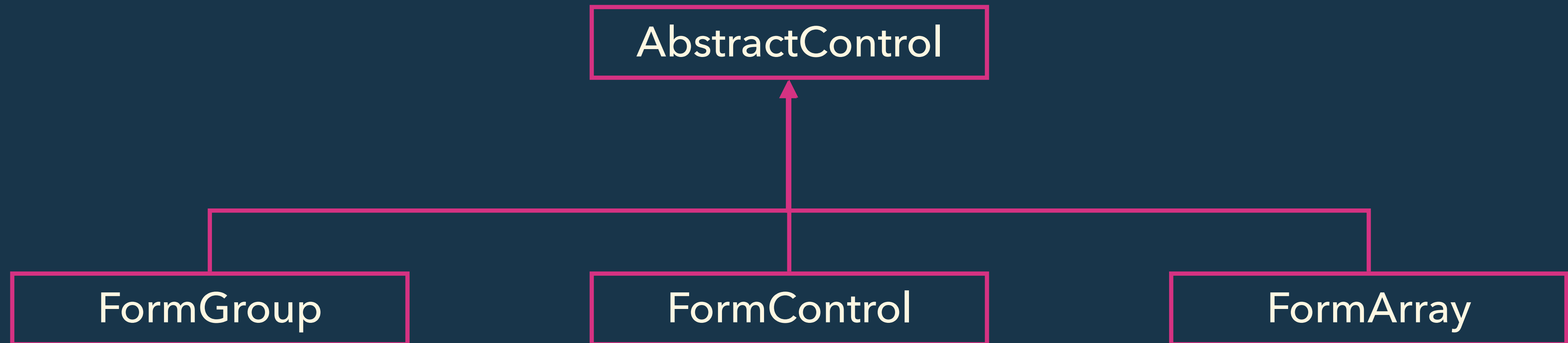
# CHARACTERISTICS

- ▶ Leverages programmatic API
- ▶ Easier to test
- ▶ Simplifies templates



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

```
@NgModule({
  declarations: [
    // ...
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
  ],
  providers: [
    // ...
  ],
  bootstrap: [
    AppComponent
  ]
})
export class AppModule { }
```



```
addNewPersonForm = new FormGroup({  
  firstName: new FormControl(),  
  lastName: new FormControl(),  
  gender: new FormControl(),  
  fav: new FormControl(),  
});
```

```
<form [formGroup]=addNewPersonForm>  
  <div class="form-group">  
    <label for="first-name">First name</label>  
    <input type="text"  
      FormControlName="firstName">  
  </div>  
  <button type="submit"  
    class="btn btn-success">Submit</button>  
</form>
```

```
addNewPersonForm = new FormGroup({  
  firstName: new FormControl(),  
  lastName: new FormControl(),  
  gender: new FormControl(),  
  fav: new FormControl(),  
});
```

```
<form [formGroup]=addNewPersonForm>  
  <div class="form-group">  
    <label for="first-name">First name</label>  
    <input type="text"  
      FormControlName="firstName">  
  </div>  
  <button type="submit"  
    class="btn btn-success">Submit</button>  
</form>
```

```
addNewPersonForm = new FormGroup({  
  firstName: new FormControl(),  
  lastName: new FormControl(),  
  gender: new FormControl(),  
  fav: new FormControl(),  
});
```

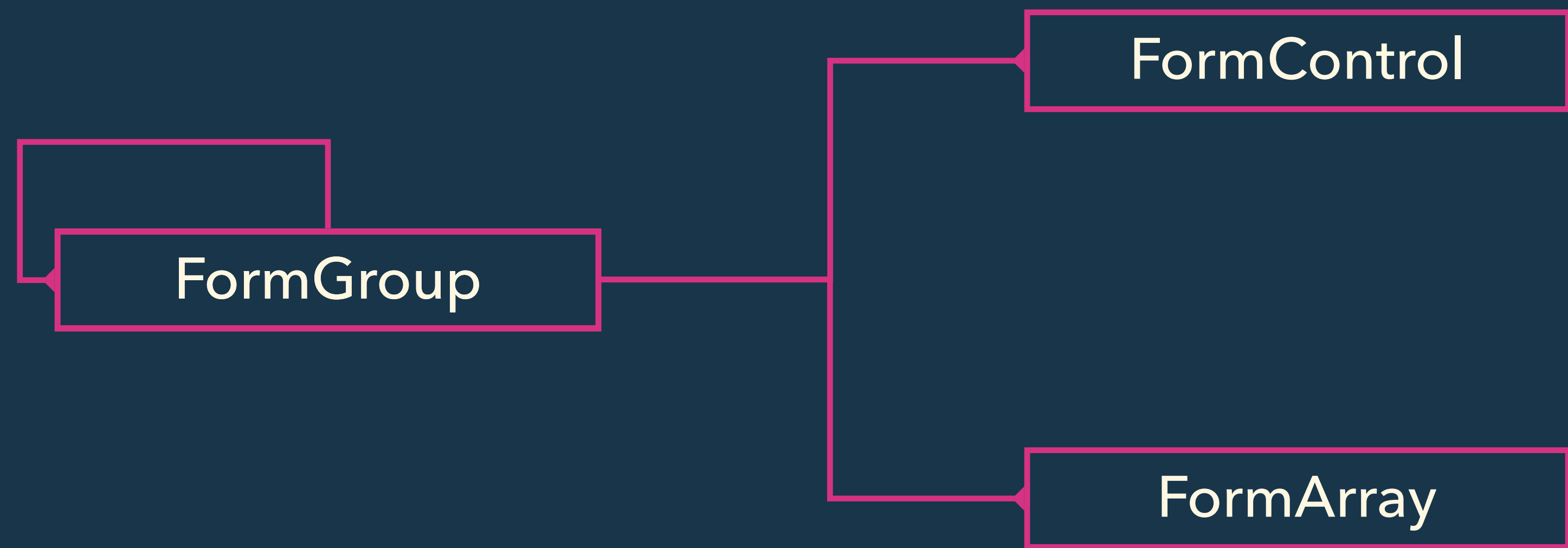
```
<form [formGroup]=addNewPersonForm>  
  <div class="form-group">  
    <label for="first-name">First name</label>  
    <input type="text"  
      FormControlName="firstName">  
  </div>  
  <button type="submit"  
    class="btn btn-success">Submit</button>  
</form>
```

```
addNewPersonForm = new FormGroup({  
  firstName: new FormControl(this.model.first_name, [Validators.required]),  
  lastName: new FormControl(this.model.last_name, [Validators.required]),  
  gender: new FormControl(this.model.gender),  
  fav: new FormControl(this.model.fav),  
});
```

```
<form [formGroup]=addNewPersonForm>  
  <div class="form-group">  
    <label for="first-name">First name</label>  
    <input type="text"  
      FormControlName="firstName">  
  </div>  
  <button [disabled]="!addNewPersonForm.valid"  
    type="submit"  
    class="btn btn-success">Submit</button>  
</form>
```

```
addNewPersonForm = new FormGroup({
  firstName: new FormControl(this.model.first_name, [Validators.required]),
  lastName: new FormControl(this.model.last_name, [Validators.required]),
  gender: new FormControl(this.model.gender),
  fav: new FormControl(this.model.fav),
});
```

```
<form [formGroup]=addNewPersonForm>
  <div class="form-group">
    <label for="first-name">First name</label>
    <input type="text"
      FormControlName="firstName">
  </div>
  <button [disabled]="!addNewPersonForm.valid"
    type="submit"
    class="btn btn-success">Submit</button>
</form>
```





**FORM BUILDER**

```
constructor(  
    private fb: FormBuilder  
) {  
    this.addNewPersonFbForm = fb.group({  
        firstName: [this.model.first_name, [Validators.required]],  
        lastName: [this.model.last_name, [Validators.required]],  
        gender: [this.model.gender],  
        fav: [this.model.fav],  
    })  
}
```

```
{
  "firstName": "Raju",
  "address": {
    "street": "1 Infinity Drive",
    "state": "CA",
    "zip": "31415"
  }
}
```

```
this.addNewPersonFbForm = fb.group({
  firstName: ['', [Validators.required]],
  address: this.fb.group({
    street: ['', ],
    state: ['', ],
    zip: ['', ],
  })
});
```

```
{  
  "firstName": "Raju",  
  "address": {  
    "street": "1 Infinity Drive",  
    "state": "CA",  
    "zip": "31415"  
  }  
}
```

```
this.addNewPersonFbForm = fb.group({  
  firstName: ['', [Validators.required]],  
  address: this.fb.group({  
    street: ['', ],  
    state: ['', ],  
    zip: ['', ],  
  })  
});
```

```
this.addNewPersonFbForm.setValue(this.model);
```

**REACTIVE?**

```
this.addNewPersonFbForm.valueChanges.forEach(  
  (value: string) => console.log(value)  
);
```

```
this.addNewPersonFbForm.get('firstName').valueChanges.forEach(  
  (value: string) => console.log(value)  
);
```

```
// Angular 5+
// Performance boost

// Single control
this.street = new FormControl(null, {
  validators: Validators.required,
  updateOn: 'blur'
});

// applies to all controls in the group
this.address = new FormGroup({
  street: new FormControl(),
  zip: new FormControl(),
  city: new FormControl()
}, { updateOn: 'submit' });
```

# REACTIVE FORMS VALIDATION



# INBUILT VALIDATORS

- ▶ Use with `Validators.<type>`
- ▶ `min`
- ▶ `max`
- ▶ `required`
- ▶ `requiredTrue`
- ▶ `email`
- ▶ `minLength`
- ▶ `maxLength`
- ▶ `pattern`
- ▶ `nullValidator`
- ▶ `compose`
- ▶ `composeAsync`

# CUSTOM VALIDATORS

### NOTES

- ▶ A simple function
  - ▶ Implement `ValidatorFn`
  - ▶ Return `null` if it passes, else return an object with an "error" key

```
export function spamValidator(control: AbstractControl) {
  let email = control.value;
  if (email && email.indexOf("@") !== -1) {
    let [, domain] = email.split("@");
    const tempEmail = [
      "guerrillamail.com",
      "getnada.com",
      "tempmailaddress.com",
    ]
    if (tempEmail.includes(domain)) {
      return {
        validateEmail: {
          valid: false
        }
      }
    }
  }
  return null;
}
```

```
// reactive forms
addNewPersonForm = new FormGroup({
  email: new FormControl(this.model.email, [spamValidator]),
});
```

```
// template forms
@Directive({
  selector: '[validDomain][ngModel]',
  providers: [
    {
      provide: NG_VALIDATORS,
      useExisting: EmailDomainValidator,
      multi: true
    }
  ]
})
class EmailDomainValidator implements Validator {
  validate(c: AbstractControl): { [key: string]: any; } {
    return spamValidator(c);
  }
}
```

```
// template forms
@NgModule({
  declarations: [
    // ...
    EmailDomainValidator,
  ],
  // ...
})
export class AppModule { }
```

```
<div class="form-group">
  <label for="email">Email</label>
  <input type="text"
    class="form-control"
    id="email"
    [(ngModel)]="model.email"
    validDomain
    name="email">
</div>
```



ALL DONE

---

**THANKS!!**

**RAJU GANDHI**