

Introduction

A form creates a cohesive, effective, and compelling data entry experience. An Angular form coordinates a set of data-bound user controls, tracks changes, validates input, and presents errors.

Form

```
<form>
  ... tags that include all
  input elements
</form>
```

All forms are placed within the HTML form tags

Standard Input Types

Text Input `<input type="text">`

Email Input `<input type="email">`

Password Input `<input type="password">`

Dropdown Selection

```
<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

Multi Selection

```
<select multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

Checkbox `<input type="checkbox">`

Radio Control `<input type="radio">`

Numeric Input `<input type="number">`

Date `<input type="date">`

Multiline Input `<textarea rows="4" cols="50"></textarea>`

Angular 2 Form - Elements (cont)

FormArray Tracks the value and validity state of an array of FormControl instances. A FormArray aggregates the values of each child FormControl into an array

FormBuilder Creates an AbstractControl from a user-specified configuration. It is essentially syntactic sugar that shortens the new FormGroup(), new FormControl(), and new FormArray() boilerplate that can build up in larger form

Requires use of FormModule

Reactive Form Names

formGroupName Used to reference a group of elements

formControlName Similar to ngModel reference to a name but simpler from a naming convention perspective

formArrayName Syncs a nested FormArray to a DOM element.

Requires the use of the ReactiveFormsModule Module

Handling Submission Event

```
<form (ngSubmit)="onSubmit()">
  ...
</form>
```

Standard Validation

Mandatory Validators.required

Minimum Length Validator.minLength(size)

Maximum Length Validators.maxLength(size)

Pattern Match Validators.pattern("regex")

Custom Validators

Displaying Validator Failures

```
<label for="name">Name</label>
  <input type="text" class="form-control" id="name" required [(ngModel)]="modelName" #name="ngModel" >
  <div [hidden]="name.valid || name.pristine" class="alert alert-danger">
    Name is required
  </div>
```

Workflow

Steps to creating a reactive form:

1. Create the Domain Model
2. Create the Controller with references to View
3. Create the View
4. Add Validations
5. Add Submit Validation Control
6. Add Dynamic Behaviors

Model

```
export interface {ModelName} {
  item?: optional;
  string | number | date | boolean
  | class | interface ([] :
  array);
}
```

Controller

FormGroup A FormGroup aggregates the values of each child FormControl into one object, with each control name as the key

FormControl Tracks the value and validation status of an individual form control. It is one of the three fundamental building blocks of Angular forms

```
function {name}(control :  
  FormControl) : {[s: string] :  
  boolean} {  
  .... function body....  
  pass return a null  
  fail return an object of type  
  {key : true}  
}
```

```
let style =  
  require('./someStyle.css');  
let template = require('./someTemplate.html');  
@Component({  
  styles : [style],  
  template: template  
});  
export class {Some}Form  
  implements OnInit{  
  
  myForm: FormGroup;  
  constructor( private fb :  
  FormBuilder) {};  
  ngOnInit() {  
    //Construct the form  
    data type  
  
    this.myForm = this.fb.  
    group({  
      'controlName' :  
      this.fb.control (...),  
      'controlArray'  
      : this.fb.array([...]),  
      'controlGroup'  
      : this.fb.group({})  
    });  
  }  
  
  onSubmit() {  
    myForm.value; //returns  
    the form values
```



By **Nathan** (Nathane2005)

cheatography.com/nathane2005/

Published 9th October, 2016.

Last updated 9th October, 2016.

Page 1 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Controller (cont)

```
> myModel = <MyModel>myForm.value;//Cast to object
}
```

Typical additions include:

1. Http Service Submission (delegate normally injected)
2. Pipes for Display customization
3. Model based Validators

View

```
<form [formGroup]='myForm'
  (ngSubmit)='onSubmit()'>
  <input formControlName='...'>
  <div formGroupName='...'>
    <input formControlName='...'>
  </div>
  <div formArrayName='...'>
    <input
      formControlName='{{index}}'
      *ngFor='let
        item of items; index = index'>
  </div>
</form>
```

Useful Blocks

```
-- Get Form Items
JSON.stringify(myForm.value)
```

Useful Links

[Angular Forms](#)
[TypeScript Basic Types](#)
[HTML Inputs](#)



By **Nathan** (Nathane2005)

cheatography.com/nathane2005/

Published 9th October, 2016.

Last updated 9th October, 2016.

Page 2 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>