

# Angular Forms

Type of Forms in Angular:

- 1)Template Driven Forms
- 2)Reactive Forms

The diagram illustrates the data flow between a user interface (UI), a model (data store), and a database. On the left, under 'Template-driven', the UI and model are connected by bidirectional dashed arrows, with a green checkmark indicating validation. A pink 'X' indicates an error state. On the right, under 'Reactive (Model-driven)', the UI and model are also connected by bidirectional dashed arrows, but the validation logic is moved to the model layer. Both diagrams show a central model layer connected to a database.

**Template-driven**

- Easy to use
- Similar to AngularJS
- Two-way data binding -> Minimal component code
- Automatically tracks form and input element state

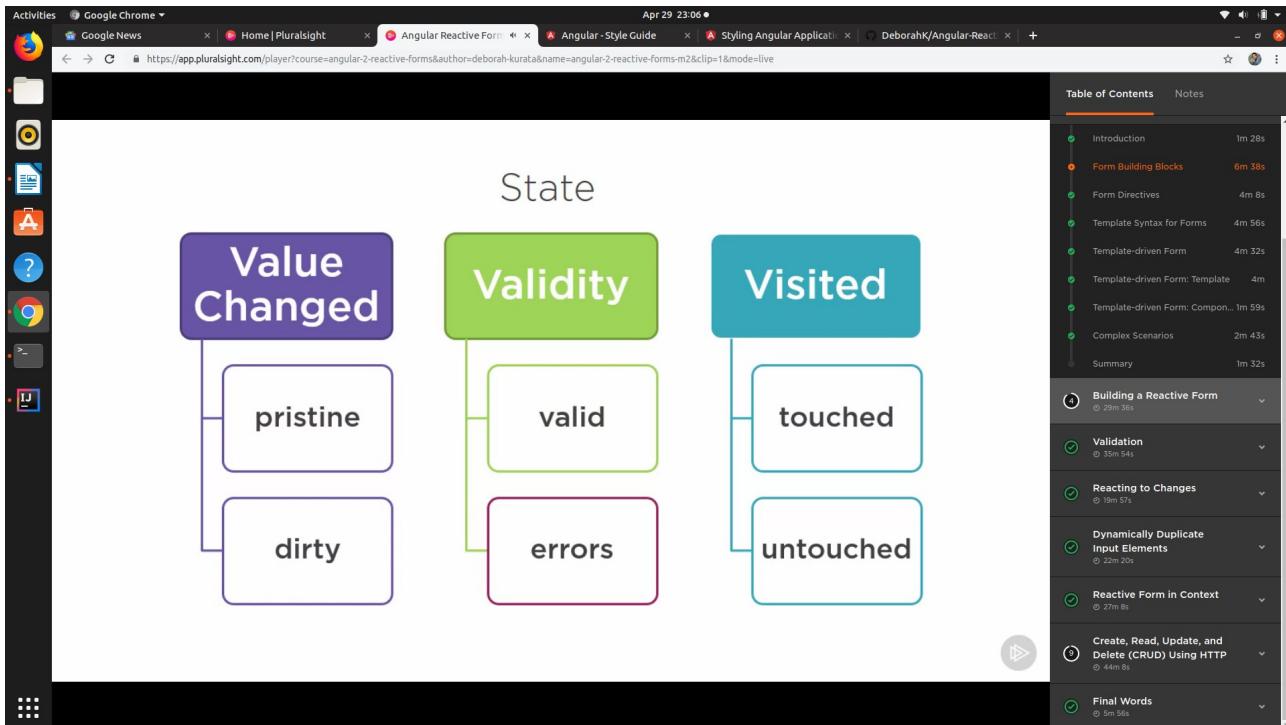
**Reactive**

- More flexible -> more complex scenarios
- Immutable data model
- Easier to perform an action on a value change
- Reactive transformations -> DebounceTime or DistinctUntilChanged
- Easily add input elements dynamically
- Easier unit testing

# Template Driven and Reactive Form

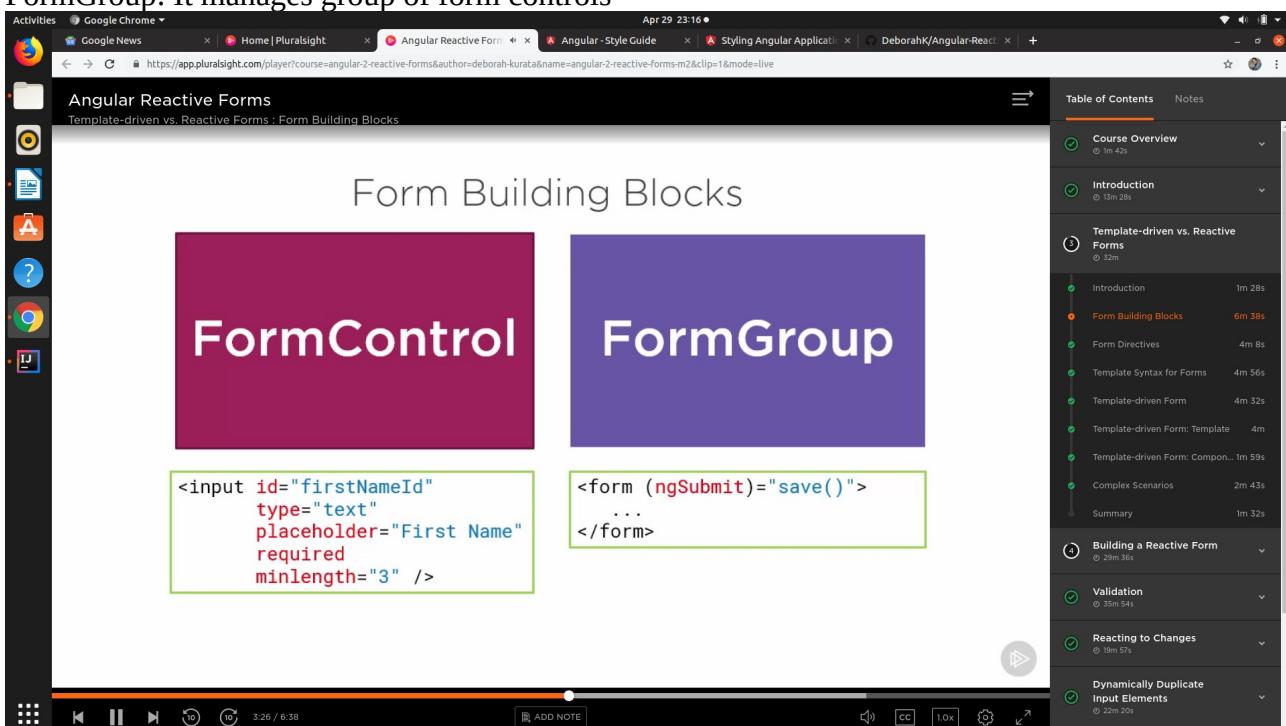
pristine: If value is unchanged the state of the value is pristine.

Dirty: Changed value state is dirty



FormControl : It tracks the state and value of an individual input element

FormGroup: It manages group of form controls



FormModel: Its a data structure that represents the html form

Angular Reactive Forms

Template-driven vs. Reactive Forms : Form Building Blocks

```

▼ controls: object
  ► email: FormControl
  ► firstName: FormControl
  ► lastName: FormControl
  ► sendCatalog: FormControl
  ► __proto__: Object
dirty: true
disabled: false
enabled: true
errors: null
invalid: false
pending: false
pristine: false
root: (...)

status: (...)

statusChanges: (...)

touched: true

untouched: false

valid: true

validator: null

▼ value: object
  email: "jack@torchwood.com"
  firstName: "Jack"
  lastName: "Harkness"
  sendCatalog: false
  ► __proto__: Object
  valueChanges: (...)
```

**Form Model**

- Retains form state
- Retains form value
- Retains child controls
  - FormControls
  - Nested FormGroups

Table of Contents

- Course Overview
- Introduction
- Template-driven vs. Reactive Forms
  - Introduction
  - Form Building Blocks
  - Form Directives
  - Template Syntax for Forms
  - Template-driven Form
  - Template-driven Form: Template
  - Template-driven Form: Component
  - Complex Scenarios
  - Summary
- Building a Reactive Form
- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements

## Template Driven Form

Template-Driven Forms

**Template**

- Form element
- Input element(s)
- Data binding
- Validation rules (attributes)
- Validation error messages
- Form model automatically generated

**Component Class**

- Properties for data binding (data model)
- Methods for form operations, such as submit

Table of Contents

- Course Overview
- Introduction
- Template-driven vs. Reactive Forms
  - Introduction
  - Form Building Blocks
  - Form Directives
  - Template Syntax for Forms
  - Template-driven Form
  - Template-driven Form: Template
  - Template-driven Form: Component
  - Complex Scenarios
  - Summary
- Building a Reactive Form
- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements

## Reactive Form

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive ...

Apr 29 23:21

<https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=1&mode=live>

## Reactive Forms Component Class

**Form model**

- Validation rules
- Validation error messages
- Properties for managing data (data model)
- Methods for form operations, such as submit

**Template**

- Form element
- Input element(s)

Binding to form model

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 1m 28s
- Template-driven vs. Reactive Forms 32m
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
  - Template-driven Form: Template 4m
  - Template-driven Form: Component 1m 59s
  - Complex Scenarios 2m 43s
  - Summary 1m 32s
- Building a Reactive Form 29m 36s
- Validation 35m 54s
- Reacting to Changes 10m 57s
- Dynamically Duplicate Input Elements 22m 20s

## Directives:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive ...

Apr 29 23:32

<https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=2&mode=live>

## Directives

**Template-driven (FormsModule)**

- ngForm
- ngModel
- ngModelGroup

```
<form (ngSubmit)="save()" #signupForm="ngForm">
  <button type="submit" [disabled]="!signupForm.valid">
    Save
  </button>
</form>
```

**FormGroup**

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 1m 28s
- Template-driven vs. Reactive Forms 32m
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
  - Template-driven Form: Template 4m
  - Template-driven Form: Component 1m 59s
  - Complex Scenarios 2m 43s
  - Summary 1m 32s
- Building a Reactive Form 29m 36s
- Validation 35m 54s
- Reacting to Changes 10m 57s
- Dynamically Duplicate Input Elements 22m 20s

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive ...

Apr 29 23:34 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=2&mode=live

## Directives

### Template-driven (FormsModule)

- ngForm
- ngModel
- ngModelGroup

```
<form (ngSubmit)="save()">
  <input id="firstNameId" type="text"
    [(ngModel)]="customer.firstName"
    name="firstName"
    #firstNameVar="ngModel"/>
</form>
```

#firstNameVar="ngModel" is highlighted with a red box.

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 1m 28s
- Template-driven vs. Reactive Forms 32m
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
  - Template-driven Form: Template 4m
  - Template-driven Form: Component 1m 59s
  - Complex Scenarios 2m 43s
  - Summary 1m 32s
- Building a Reactive Form 29m 36s
  - Validation 35m 54s
  - Reacting to Changes 10m 57s
  - Dynamically Duplicate Input Elements 22m 20s

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive ...

Apr 29 23:35 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=2&mode=live

## Directives

### Template-driven (FormsModule)

- ngForm
- ngModel
- ngModelGroup

### Reactive (ReactiveFormsModule)

- formGroup
- formControl
- formControlName
- formGroupName
- formArrayName

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 1m 28s
- Template-driven vs. Reactive Forms 32m
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
  - Template-driven Form: Template 4m
  - Template-driven Form: Component 1m 59s
  - Complex Scenarios 2m 43s
  - Summary 1m 32s
- Building a Reactive Form 29m 36s
  - Validation 35m 54s
  - Reacting to Changes 10m 57s
  - Dynamically Duplicate Input Elements 22m 20s

Template Driven Form Template:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive Apr 29 23:41 •

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=3&mode=live

## Template-driven Form

```
customer.component.html
```

```
<form (ngSubmit)="save()">
  <div>
    <label for="firstNameId">First Name</label>
    <input id="firstNameId" type="text"
      placeholder="First Name (required)"
      required
      minlength="3"
      [(ngModel)]="customer.firstName"
      name="firstName"
      #firstNameVar="ngModel"
      [ngClass]="{'is-invalid': firstNameVar.touched && !firstNameVar.valid }" />
    <span *ngIf="firstNameVar.errors">
      Please enter your first name.
    </span>
  ...
  <button type="submit">Save</button>
</form>
```

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 10m 28s
- Template-driven vs. Reactive Forms 32m
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
    - Template-driven Form: Template 4m
    - Template-driven Form: Component 1m 59s
  - Complex Scenarios 2m 43s
  - Summary 1m 32s
- Building a Reactive Form 29m 36s
  - Validation 35m 54s
  - Reacting to Changes 10m 57s
  - Dynamically Duplicate Input Elements 22m 20s

### Reactive Form Template;

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive Apr 29 23:49 •

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=3&mode=live

## Reactive Form

```
customer.component.html
```

```
<form (ngSubmit)="save()" [FormGroup]="signupForm">
  <div>
    <label for="firstNameId">First Name</label>
    <input id="firstNameId" type="text"
      placeholder="First Name (required)"
      formControlName="firstName"
      [ngClass]="{'is-invalid': formError.firstName }" />
    <span *ngIf="formError.firstName">
      {{formError.firstName}}
    </span>
  ...
  <button type="submit">Save</button>
</form>
```

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 10m 28s
- Template-driven vs. Reactive Forms 32m
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
    - Template-driven Form: Template 4m
    - Template-driven Form: Component 1m 59s
  - Complex Scenarios 2m 43s
  - Summary 1m 32s
- Building a Reactive Form 29m 36s
  - Validation 35m 54s
  - Reacting to Changes 10m 57s
  - Dynamically Duplicate Input Elements 22m 20s

Scenario in which Template Driven Approach becomes difficult to use

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive-Form

Apr 30 00:11 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=7&mode=live

## Complex Scenarios



- Dynamically add input elements
- Watch what the user types
- Wait validation until typing stops
- Different validation for different situations
- Immutable data structures

Table of Contents Notes

- Forms
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
  - Template-driven Form: Template 4m
  - Template-driven Form: Component 1m 59s
  - Complex Scenarios** 2m 43s
  - Summary 1m 32s
- Building a Reactive Form
  - Introduction 1m 19s
  - The Component Class 3m 52s
  - The Component Class: Demo 4m 28s
  - The Angular Module 1m 31s
  - The Template 3m 29s
  - The Template: Demo 4m 19s
  - Using setValue and patchValue 2m 26s
  - Simplifying with FormBuilder 4m 48s
  - Checklists and Summary 3m 20s
- Validation

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive-Form

Apr 30 00:13 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m2&clip=8&mode=live

## Angular Forms

Template-driven	Reactive
Generated form model	Manually created form model
HTML validation	Validation in the class
Two-way data binding	No two-way data binding

Table of Contents Notes

- Forms
  - Introduction 1m 28s
  - Form Building Blocks 6m 38s
  - Form Directives 4m 8s
  - Template Syntax for Forms 4m 56s
  - Template-driven Form 4m 32s
  - Template-driven Form: Template 4m
  - Template-driven Form: Component 1m 59s
  - Complex Scenarios** 2m 43s
  - Summary 1m 32s
- Building a Reactive Form
  - Introduction 1m 19s
  - The Component Class 3m 52s
  - The Component Class: Demo 4m 28s
  - The Angular Module 1m 31s
  - The Template 3m 29s
  - The Template: Demo 4m 19s
  - Using setValue and patchValue 2m 26s
  - Simplifying with FormBuilder 4m 48s
  - Checklists and Summary 3m 20s
- Validation

## Building a Reactive Form

Reactive form must have at least one form group

**Form Model**

- Root FormGroup
- FormControl for each input element
- Nested FormGroups as desired
- FormArrays

Creating Form Group: Multiple FormControl s are added to the FormGroup. We can provide the default value to the FormControl along with the validations.

```

customer.component.ts

...
import { FormGroup, FormControl } from '@angular/forms';

...
export class CustomerComponent implements OnInit {
  ...
  ngOnInit(): void {
    this.customerForm = new FormGroup({
      firstName: new FormControl(),
      lastName: new FormControl(),
      email: new FormControl(),
      sendCatalog: new FormControl(true)
    });
  }
}

```

Note: We could have created the instance of FormGroup in constructor instead of ngOnInit but we selected the ngOnInit to ensure that component and template are initialized before building the Form model

We need below fields for the reactive form

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive ...

Apr 30 00:40 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=4&mode=live

Angular Reactive Forms Building a Reactive Form : The Template

## Reactive Forms Directives

### Reactive Forms

- **formGroup**
- **formControl**
- **formControlName**
- **formGroupName**
- **formArrayName**

Table of Contents Notes

- Reactive Forms 32m
  - Building a Reactive Form 29m 36s
    - Introduction 1m 19s
    - The Component Class 3m 52s
    - The Component Class: Demo 4m 28s
    - The Angular Module 1m 31s
    - The Template 3m 29s
      - The Template: Demo 4m 19s
      - Using setValue and patchValue 2m 26s
      - Simplifying with FormBuilder 4m 48s
    - Checklists and Summary 3m 20s
  - Validation 35m 54s
  - Reacting to Changes 19m 57s
  - Dynamically Duplicate Input Elements 22m 20s
  - Reactive Form in Context 27m 8s
  - Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
    - Final Words

FormGroup and FormControlName are mapped as shown in the html

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive ...

Apr 30 00:42 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=4&mode=live

Angular Reactive Forms Building a Reactive Form : The Template

## formControlName

### customer.component.html

```
<form (ngSubmit)="save()" [formGroup]="customerForm">
  <div>
    <label for="firstNameId">First Name</label>
    <input id="firstNameId" type="text"
      placeholder="First Name (required)"
      formControlName="firstName" />
    <span ... >
      ...
    </span>
  </div>
  ...
</form>
```

Table of Contents Notes

- Reactive Forms 32m
  - Building a Reactive Form 29m 36s
    - Introduction 1m 19s
    - The Component Class 3m 52s
    - The Component Class: Demo 4m 28s
    - The Angular Module 1m 31s
    - The Template 3m 29s
      - The Template: Demo 4m 19s
      - Using setValue and patchValue 2m 26s
      - Simplifying with FormBuilder 4m 48s
    - Checklists and Summary 3m 20s
  - Validation 35m 54s
  - Reacting to Changes 19m 57s
  - Dynamically Duplicate Input Elements 22m 20s
  - Reactive Form in Context 27m 8s
  - Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
    - Final Words

The screenshot shows a Pluralsight video player interface. On the left, there's a sidebar with various icons. The main content area has a title "Checklist: Template". Below it, two sections of code are shown in boxes:

- Bind the form element to the FormGroup property**

```
<form (ngSubmit)="save()" [formGroup]="customerForm">
```

On the right, there's a "Table of Contents" sidebar with several sections listed:

- Reactive Forms (selected)
- Building a Reactive Form
  - Introduction
  - The Component Class
  - The Component Class: Demo
  - The Angular Module
  - The Template
  - The Template: Demo
  - Using setValue and patchValue
  - Simplifying with FormBuilder
  - Checklists and Summary
- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Final Words

FormModel properties can be accessed using the below properties

The screenshot shows a Pluralsight video player interface. On the left, there's a sidebar with various icons. The main content area has a title "Accessing the Form Model Properties". Below it, several examples of code are shown in boxes:

- `customerForm.controls.firstName.valid`
- `customerForm.get('firstName').valid`
- `firstName = new FormControl();`
- `ngOnInit(): void {  
 this.customerForm = new FormGroup({  
 firstName: this.firstName,  
 ...  
 });  
}`
- `firstName.valid`

A red arrow points from the word "firstName" in the fourth example up to the declaration of `firstName` in the fifth example. On the right, there's a "Table of Contents" sidebar with several sections listed:

- Reactive Forms (selected)
- Building a Reactive Form
  - Introduction
  - The Component Class
  - The Component Class: Demo
  - The Angular Module
  - The Template
  - The Template: Demo
  - Using setValue and patchValue
  - Simplifying with FormBuilder
  - Checklists and Summary
- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Final Words

For setting default values to the Form Attributes in Reactive form we use `setValue` and `patchValue`. `SetValue` if for providing default values to all the fields and `patchValue` for subset of the values.

The screenshot shows a Pluralsight video player interface. The top navigation bar includes 'Activities', 'Google Chrome', 'Google News', 'Home | Pluralsight', 'Angular Reactive Form', 'Angular - Style Guide', 'Styling Angular Application', 'DeborahK/Angular-React', 'Demo', and a search bar. The main content area displays a slide titled 'Using setValue and patchValue' with two code snippets:

```
this.customerForm.setValue({  
  firstName: 'Jack',  
  lastName: 'Harkness',  
  email: 'jack@torchwood.com'  
});
```

```
this.customerForm.patchValue({  
  firstName: 'Jack',  
  lastName: 'Harkness'  
});
```

To the right is a sidebar with a 'Table of Contents' and 'Notes' section. The 'Table of Contents' lists chapters for 'Building a Reactive Form' (Introduction, The Component Class, The Angular Module, The Template, The Template: Demo), 'Validation' (Checklists and Summary), 'Reacting to Changes' (Validation), 'Dynamically Duplicate Input Elements' (Delete (CRUD) Using HTTP), and 'Reactive Form in Context'. The 'Notes' section is currently empty.

## Example

The screenshot shows a Microsoft Edge browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=6&mode=live>. The page title is "Angular Reactive Forms". The main content is a code editor with the file `customer.component.ts` open. The code demonstrates building a reactive form using `patchValue` and `setValue`. A red box highlights the `this.customerForm.patchValue({})` call at line 27. The browser's status bar shows "Ln 27, Col 28". On the right side of the browser, there is a sidebar with a "Table of Contents" and a "Notes" section, listing various chapters and sections of the course.

```
customer.component.ts
1 import { Component } from '@angular/core';
2 import { FormBuilder, FormGroup, FormControl } from '@angular/forms';
3 import { Customer } from './customer';
4
5 @Component({
6   selector: 'app-customer',
7   templateUrl: './customer.component.html',
8   styleUrls: ['./customer.component.css']
9 })
10
11 export class CustomerComponent implements OnInit {
12   customerForm: FormGroup;
13   customer = new Customer();
14
15   constructor() { }
16
17   ngOnInit() {
18     this.customerForm = new FormGroup({
19       firstName: new FormControl(),
20       lastName: new FormControl(),
21       email: new FormControl(),
22       sendCatalog: new FormControl(true)
23     });
24   }
25
26   populateTestData(): void {
27     this.customerForm.patchValue({
28       firstName: 'Jack',
29       lastName: 'Harkness',
30       sendCatalog: false
31     });
32   }
33
34   save() {
35     console.log(this.customerForm);
36   }
37 }
```

**FormBuilder:** It is a class that can be used when creating Reactive Forms. It is like a factory that creates formgroups and formcontrols for us.

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive Demo

Apr 30 01:15

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=7&mode=live

# FormBuilder



**Creates a form model from a configuration**

**Shortens boilerplate code**

**Provided as a service**

Table of Contents Notes

- Reactive Forms 32m
  - Building a Reactive Form 29m 36s
    - Introduction 1m 19s
    - The Component Class 3m 52s
    - The Component Class: Demo 4m 28s
    - The Angular Module 1m 31s
    - The Template 3m 29s
    - The Template: Demo 4m 19s
    - Using setValue and patchValue 2m 26s
    - Simplifying with FormBuilder 4m 48s
    - Checklists and Summary 3m 20s
  - Validation 35m 54s
  - Reacting to Changes 19m 57s
  - Dynamically Duplicate Input Elements 22m 20s
  - Reactive Form in Context 27m 8s
  - Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Final Words

Play

Steps for creating a form builder (It is provided as a service so it is implemented in the same way)

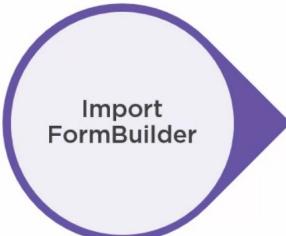
Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive Demo

Apr 30 01:16

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=7&mode=live

# FormBuilder Steps



```
import { FormBuilder } from '@angular/forms';
```

Table of Contents Notes

- Reactive Forms 32m
  - Building a Reactive Form 29m 36s
    - Introduction 1m 19s
    - The Component Class 3m 52s
    - The Component Class: Demo 4m 28s
    - The Angular Module 1m 31s
    - The Template 3m 29s
    - The Template: Demo 4m 19s
    - Using setValue and patchValue 2m 26s
    - Simplifying with FormBuilder 4m 48s
    - Checklists and Summary 3m 20s
  - Validation 35m 54s
  - Reacting to Changes 19m 57s
  - Dynamically Duplicate Input Elements 22m 20s
  - Reactive Form in Context 27m 8s
  - Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Final Words

Play

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Forms Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive Forms Demo

Apr 30 01:17 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=7&mode=live

## FormBuilder Steps

The diagram consists of two overlapping circles. The left circle is purple and contains the text "Import FormBuilder". The right circle is green and contains the text "Inject the FormBuilder instance". A large purple arrow points from the left circle to the right circle.

```
constructor(private fb: FormBuilder) { }
```

Table of Contents Notes

- Reactive Forms 32m
  - Building a Reactive Form 29m 36s
    - Introduction 1m 19s
    - The Component Class 3m 52s
    - The Component Class: Demo 4m 28s
    - The Angular Module 1m 31s
    - The Template 3m 29s
    - The Template: Demo 4m 19s
    - Using setValue and patchValue 2m 26s
    - Simplifying with FormBuilder 4m 48s
    - Checklists and Summary 3m 20s
  - Validation 35m 54s
  - Reacting to Changes 18m 57s
  - Dynamically Duplicate Input Elements 22m 20s
  - Reactive Form in Context 27m 8s
  - Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Final Words

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form Angular - Style Guide Styling Angular Application DeborahK/Angular-Reactive Forms Demo

Apr 30 01:17 • https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m3&clip=7&mode=live

## FormBuilder Steps

The diagram consists of three overlapping circles. The left circle is purple and contains the text "Import FormBuilder". The middle circle is green and contains the text "Inject the FormBuilder instance". The right circle is light green and contains the text "Use the instance". A large purple arrow points from the left circle to the middle circle, and a green arrow points from the middle circle to the right circle.

```
this.customerForm = this.fb.group({  
  firstName: null,  
  lastName: null,  
  email: null,  
  sendCatalog: true  
});
```

Table of Contents Notes

- Reactive Forms 32m
  - Building a Reactive Form 29m 36s
    - Introduction 1m 19s
    - The Component Class 3m 52s
    - The Component Class: Demo 4m 28s
    - The Angular Module 1m 31s
    - The Template 3m 29s
    - The Template: Demo 4m 19s
    - Using setValue and patchValue 2m 26s
    - Simplifying with FormBuilder 4m 48s
    - Checklists and Summary 3m 20s
  - Validation 35m 54s
  - Reacting to Changes 18m 57s
  - Dynamically Duplicate Input Elements 22m 20s
  - Reactive Form in Context 27m 8s
  - Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Final Words

Syntax:

The screenshot shows a Pluralsight video player interface. The main content area displays three snippets of Angular code related to FormBuilder's FormControl syntax. The first snippet shows a basic group definition with a single field. The second snippet adds a validation rule ('n/a') and a disabled state. The third snippet uses an array to define multiple fields simultaneously. The right side of the screen features a sidebar with a 'Table of Contents' section listing various video chapters on reactive forms, validation, and other topics.

```
this.customerForm = this.fb.group({
  firstName: '',
  sendCatalog: true
});
```

```
this.customerForm = this.fb.group({
  firstName: {value: 'n/a', disabled: true},
  sendCatalog: {value: true, disabled: false}
});
```

```
this.customerForm = this.fb.group([
  {firstName: ['', Validators.required], sendCatalog: true}
]);
```

## Validation

As shown below in the firstName feild first “” hold the value and then we have array of validators. The arrow in the image below is for the async validator which is used during async calls.Example of asynch calls is calling a serverside validation method to minimize the async calls, note that all asynch validators are not executed until all synchronous validators are executed

The screenshot shows a Pluralsight video player interface. The main content area displays two snippets of Angular code for setting built-in validation rules. The first snippet shows a basic group with a required field. The second snippet adds a minLength validator to the same field. The right side of the screen features a sidebar with a 'Table of Contents' section listing various video chapters on reactive forms, validation, and other topics.

```
this.customerForm = this.fb.group({
  firstName: ['', Validators.required],
  sendCatalog: true
});
```

```
this.customerForm = this.fb.group({
  firstName: ['', Validators.required, Validators.minLength(3)]▲,
  sendCatalog: true
});
```

We can change the validation for any form control in runtime using **setValidators()** method on that form control and pass in the new validator.

**clearValidators()**: Remove validators method it can be used in certain circumstances where we do not need validator.

**updateValueAndValidity()**: If we change the validator and then want values to be re-evaluated we need to call this method on that form control as well.

The screenshot shows a browser window with several tabs open. The main content area displays four code snippets:

```
myControl.setValidators(Validators.required);
```

```
myControl.setValidators([Validators.required,  
                         Validators.maxLength(30)]);
```

```
myControl.clearValidators();
```

```
myControl.updateValueAndValidity();
```

To the right of the browser is a sidebar titled "Table of Contents" which lists various sections of the course, such as "Course Overview", "Introduction", "Template-driven vs. Reactive Forms", "Building a Reactive Form", "Validation", and "Reacting to Changes".

If there is a requirement to make phone/email based of the notification we have to do below changes.

## 1) Add code for the phone field and notification fields

The screenshot shows a Visual Studio Code interface with a file named "customer.component.html" open. A specific section of the code is highlighted with a red box:

```
<div class="form-group row mb-2">  
  <label class="col-md-2 col-form-label" for="phoneId">Phone</label>  
  <div class="col-md-8">  
    <input class="form-control" id="phoneId" type="tel" placeholder="Phone" formControlName="phone"  
          [ngClass]="'is-invalid': !customerForm.get('phone').valid }" />  
    <span class="invalid-feedback">  
      <span *ngIf="customerForm.get('phone').errors?.required">  
        Please enter your phone number.  
      </span>  
    </span>  
  </div>  
</div>
```

The code implements a reactive form control for a phone number, using Angular's built-in validators and form control properties like `Validators.required` and `Validators.maxLength`. It also uses the `ngClass` directive to apply a CSS class to the input field if it is invalid.

The screenshot shows a browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=2&mode=live>. The page title is "Angular Reactive Forms". The main content area displays code for "customer.component.html" and "customer.component.ts". The code in "customer.component.html" is highlighted with a red box around the following section:

```
<div class="form-group row mb-2">
  <label class="col-md-2 col-form-label pt-0">Send Notifications</label>
  <div class="col-md-8">
    <div class="form-check form-check-inline">
      <label class="form-check-label">
        <input class="form-check-input" type="radio" value="email" formControlName="notification"> Email
      </label>
    </div>
    <div class="form-check form-check-inline">
      <label class="form-check-label">
        <input class="form-check-input" type="radio" value="text" formControlName="notification"> Text
      </label>
    </div>
  </div>
</div>
```

The "customer.component.ts" file shows the corresponding TypeScript code:

```
import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-customer',
  templateUrl: './customer.component.html',
  styleUrls: ['./customer.component.css']
})
export class CustomerComponent {
  customerForm: FormGroup;

  constructor(private fb: FormBuilder) {
    this.customerForm = this.fb.group({
      name: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      notification: ''
    });
  }
}
```

The right side of the screen features a sidebar with a "Table of Contents" and a "Notes" section. The table of contents lists various chapters and their durations:

- Course Overview (1m 42s)
- Introduction (1m 26s)
- Template-driven vs. Reactive Forms (32m)
- Building a Reactive Form (29m 36s)
- Validation (5m 54s)
  - Introduction (1m 58s)
  - Setting Built-in Validation Rules (4m 43s)
  - Adjusting Validation Rules at Runtime (5m 27s)
  - Custom Validators (5m 35s)
  - Custom Validation with Parameters (3m 6s)
  - Cross-Field Validation: Nested Form Groups (4m 58s)
  - Cross-field Validation: Custom Validation Function (5m 16s)
  - Checklists and Summary (4m 47s)
- Reacting to Changes (19m 57s)
- Dynamically Duplicate Input Elements (22m 20s)
- Reactive Form in Context (27m 56s)

Add fields in component

The screenshot shows a browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=2&mode=live>. The page title is "Angular Reactive Forms". The main content area displays code for a Customer component:

```
customer.component.ts
1 import { Component, OnInit } from '@angular/core';
2 import { FormGroup, FormBuilder, Validators } from '@angular/forms';
3
4 import { Customer } from './customer';
5
6 @Component({
7   selector: 'app-customer',
8   templateUrl: './customer.component.html',
9   styleUrls: ['./customer.component.css']
10 })
11 export class CustomerComponent implements OnInit {
12   customerForm: FormGroup;
13   customer = new Customer();
14
15   constructor(private fb: FormBuilder) { }
16
17   ngOnInit() {
18     this.customerForm = this.fb.group({
19       firstName: ['', [Validators.required, Validators.minLength(3)]],
20       lastName: ['', [Validators.required, Validators.maxLength(50)]],
21       email: ['', [Validators.required, Validators.email]],
22       phone: '',
23       notification: 'email',
24       sendCatalog: true
25     });
26   }
27 }
```

The code editor highlights several validation rules in red boxes: `Validators.required` and `Validators.minLength(3)` for `firstName`, `Validators.required` and `Validators.maxLength(50)` for `lastName`, `Validators.required` and `Validators.email` for `email`, and `Validators.required` for `notification`.

The right sidebar contains a "Table of Contents" and "Notes" section, listing various chapters of the course:

- Course Overview
- Introduction
- Template-driven vs. Reactive Forms
- Building a Reactive Form
- Validation
  - Introduction
  - Setting Built-in Validation Rules
  - Adjusting Validation Rules at Runtime
  - Custom Validators
  - Custom Validation with Parameters
  - Cross-Field Validation: Nested Forms
  - Cross-field Validation: Custom Logic
  - Checklists and Summary
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Directive in Context

## Template image

As based on the selection of the radio button phone/email becomes mandatory so the validation will be based on the radio button. To achieve that we require a method

```

<div class="form-group row mb-2">
  <label class="col-md-2 col-form-label pt-0">Send Notifications</label>
  <div class="col-md-8">
    <div class="form-check form-check-inline">
      <label class="form-check-label">
        <input class="form-check-input" type="radio" value="email" formControlName="notification" (click)="setNotification('email')"/> Email
      </label>
    </div>
    <div class="form-check form-check-inline">
      <label class="form-check-label">
        <input class="form-check-input" type="radio" value="text" formControlName="notification" (click)="setNotification('text')"/> Text
      </label>
    </div>
  </div>
</div>

```

Method Code in the component

```

    24     sendCatalog: true
    25   );
    26 }
    27
    28 populateTestData(): void {
    29   this.customerForm.patchValue({
    30     firstName: 'Jack',
    31     lastName: 'Harkness',
    32     sendCatalog: false
    33   });
    34 }
    35
    36 save() {
    37   console.log(this.customerForm);
    38   console.log(`Saved: ${JSON.stringify(this.customerForm.value)}`);
    39 }
    40
    41 setNotification(notifyVia: string): void {
    42   const phoneControl = this.customerForm.get('phone');
    43   if (notifyVia === 'text') {
    44     phoneControl.setValidators([Validators.required]);
    45   } else {
    46     phoneControl.clearValidators();
    47   }
    48   phoneControl.updateValueAndValidity();
    49 }
  
```

In 48, Col 43 | Spaces: 2 | UTF-8 | LF | TypeScript 3.0.3 | 0:00 0:00 0:00

Table of Contents Notes

- Course Overview
- Introduction
- Template-driven vs. Reactive Forms
- Building a Reactive Form
- Validation**
- Introduction
- Setting Built-in Validation Rules
- Adjusting Validation Rules at Runtime
- Custom Validators
- Custom Validation with Parameters
- Cross-Field Validation: Nested ...
- Cross-field Validation: Custom ...
- Checklists and Summary
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context

## Custom Validator: Abstract control takes formgroup of formcontrol

```

function myValidator(c: AbstractControl): {[key: string]: boolean} | null {
  if (somethingIsWrong) {
    return { 'myvalidator': true };
  }
  return null;
}
  
```

1:16 / 5:35 | ADD NOTE | 0:00 0:00 0:00

Table of Contents Notes

- Reactive Forms
- Building a Reactive Form
- Validation**
- Introduction
- Setting Built-in Validation Rules
- Adjusting Validation Rules at Runtime
- Custom Validators**
- Custom Validation with Parameters
- Cross-Field Validation: Nested ...
- Cross-field Validation: Custom ...
- Checklists and Summary
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Final Words

Example:  
Html template

The screenshot shows a browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=3&mode=live>. The page title is "Angular Reactive Forms". The main content area displays code for a rating form control. A red box highlights the validation logic for the rating input:

```
<div class="form-group row mb-2">
  <label class="col-md-2 col-form-label" for="ratingId">Rating</label>
  <div class="col-md-8">
    <input class="form-control" id="ratingId" type="number" formControlName="rating" [ngClass]="{{'is-invalid': (customerForm.get('rating').touched || customerForm.get('rating').dirty) && !customerForm.get('rating').valid }}"/>
    <span class="invalid-feedback">
      <span *ngIf="customerForm.get('rating').errors?.range">
        Please rate your experience from 1 to 5.
      </span>
    </span>
  </div>
</div>
```

The right side of the screen features a sidebar with a "Table of Contents" and "Notes" section, listing various chapters and their durations. The "Custom Validators" chapter is currently selected.

Then we create a method, since this method will only be used by the CustomerComponent so we are keeping it above the class as shown below.

The screenshot shows a Google Chrome browser window with the following tabs:

- Activities
- Google News
- Home | Pluralsight
- Angular Reactive Form
- Demo

The URL is <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=3&mode=live>. The page title is "May 1 23:56 •".

The main content area displays the `customer.component.ts` file from a Visual Studio Code editor. The code implements an Angular reactive form for a customer. It includes a custom validator for rating ranges and handles the creation of a new customer.

```
customer.component.ts - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
customer.component.html TS customer.component.ts x

1 import { Component, OnInit } from '@angular/core';
2 import { FormGroup, FormBuilder, Validators } from '@angular/forms';
3
4 import { Customer } from './customer';
5
6 function ratingRange(c: AbstractControl) { [key: string]: boolean } | null {
7   if (c.value != null && (!isNaN(c.value) || c.value < 1 || c.value > 5)) {
8     return { 'range': true };
9   }
10  return null;
11 }
12
13 @Component({
14   selector: 'app-customer',
15   templateUrl: './customer.component.html',
16   styleUrls: ['./customer.component.css']
17 })
18 export class CustomerComponent implements OnInit {
19   customerForm: FormGroup;
20   customer = new Customer();
21
22   constructor(private fb: FormBuilder) { }
23
24   ngOnInit() {
25     this.customerForm = this.fb.group({
26       firstName: ['', [Validators.required, Validators.minLength(3)]],
27       lastName: ['', [Validators.required, Validators.maxLength(50)]]
28     });
29   }
30 }

Ln 17, Col 3  Spaces: 2  UTF-8  LF  TypeScript 3.0.3  ⚡  📢  🔔
```

The right side of the screen features a sidebar with a "Table of Contents" and a "Notes" section. The table of contents lists various chapters and their durations:

- Reactive Forms (32m)
- Building a Reactive Form (29m 36s)
- Validation (35m 54s)
- Introduction (1m 58s)
- Setting Built-in Validation Rules (4m 43s)
- Adjusting Validation Rules at R... (5m 27s)
- Custom Validators (5m 35s)
- Custom Validation with Paramet... (3m 6s)
- Cross-Field Validation: Nested ... (4m 58s)
- Cross-field Validation: Custom ... (5m 16s)
- Checklists and Summary (4m 47s)
- Reacting to Changes (10m 57s)
- Dynamically Duplicate Input Elements (22m 20s)
- Reactive Form in Context (27m 8s)
- Create, Read, Update, and Delete (CRUD) Using HTTP (44m 8s)
- Final Words (5m 56s)

```

    6  function ratingRange(c: AbstractControl): {[key: string]: boolean} | null {
    7    if (c.value !== null && isNaN(c.value) || c.value < 1 || c.value > 5) {
    8      return { 'range': true };
    9    }
   10   return null;
   11 }
   12
   13 @Component({
   14   selector: 'app-customer',
   15   templateUrl: './customer.component.html',
   16   styleUrls: ['./customer.component.css']
   17 })
   18 export class CustomerComponent implements OnInit {
   19   customerForm: FormGroup;
   20   customer = new Customer();
   21
   22   constructor(private fb: FormBuilder) { }
   23
   24   ngOnInit() {
   25     this.customerForm = this.fb.group({
   26       firstName: ['', [Validators.required, Validators.minLength(3)]],
   27       lastName: ['', [Validators.required, Validators.maxLength(50)]],
   28       email: ['', [Validators.required, Validators.email]],
   29       phone: '',
   30       notification: 'email',
   31       rating: [null, ratingRange],
   32     });
  
```

In 31, Col 33 Spaces: 2 UTF-8 LF TypeScript 3.0.3

Table of Contents Notes

- Reactive Forms (32m)
- Building a Reactive Form (29m 36s)
- Validation (35m 54s)
  - Introduction (1m 58s)
  - Setting Built-in Validation Rules (4m 43s)
  - Adjusting Validation Rules at Runtime (5m 27s)
  - Custom Validators (5m 35s)
  - Custom Validation with Parameters (3m 6s)
  - Cross-Field Validation: Nested Forms (4m 58s)
  - Cross-Field Validation: Custom Validators (5m 16s)
  - Checklists and Summary (4m 47s)
- Reacting to Changes (19m 57s)
- Dynamically Duplicate Input Elements (22m 20s)
- Reactive Form in Context (27m 8s)
- Create, Read, Update, and Delete (CRUD) Using HTTP (44m 8s)
- Final Words (5m 56s)

**Custom Validator with Parameters:** In this scenario we need to create factory function with return type as shown below

## Custom Validator with Parameters

```

function myValidator(param: any): ValidatorFn {
  return (c: AbstractControl): {[key: string]: boolean} | null => {
    if (somethingIsWrong) {
      return { 'myvalidator': true };
    }
    return null;
  };
}

```

In 31, Col 33 Spaces: 2 UTF-8 LF TypeScript 3.0.3

Table of Contents Notes

- Reactive Forms (32m)
- Building a Reactive Form (29m 36s)
- Validation (35m 54s)
  - Introduction (1m 58s)
  - Setting Built-in Validation Rules (4m 43s)
  - Adjusting Validation Rules at Runtime (5m 27s)
  - Custom Validators (5m 35s)
  - Custom Validation with Parameters (3m 6s)
  - Cross-Field Validation: Nested Forms (4m 58s)
  - Cross-Field Validation: Custom Validators (5m 16s)
  - Checklists and Summary (4m 47s)
- Reacting to Changes (19m 57s)
- Dynamically Duplicate Input Elements (22m 20s)
- Reactive Form in Context (27m 8s)
- Create, Read, Update, and Delete (CRUD) Using HTTP (44m 8s)
- Final Words (5m 56s)

```

    import { Component, OnInit } from '@angular/core';
    import { FormGroup, FormBuilder, Validators, AbstractControl, ValidatorFn } from '@angular/forms';
    import { Customer } from './customer';

    function ratingRange(min: number, max: number): ValidatorFn {
      return (c: AbstractControl): { [key: string]: boolean } | null => {
        if (c.value !== null && (!isNaN(c.value) || c.value < min || c.value > max)) {
          return { 'range': true };
        }
        return null;
      };
    }

    @Component({
      selector: 'app-customer',
      templateUrl: './customer.component.html',
      styleUrls: ['./customer.component.css']
    })
    export class CustomerComponent implements OnInit {
      customerForm: FormGroup;
      customer = new Customer();

      constructor(private fb: FormBuilder) { }

      ngOnInit() {
        this.customerForm = this.fb.group({

```

Now we have added the function as shown above , we are passing the two parameet rather making max, min as fixed 1 and 5 in the function discussed earlier.

We will remove the function highlighted above with function nake and wil add => as shown below

```

    import { Component, OnInit } from '@angular/core';
    import { FormGroup, FormBuilder, Validators, AbstractControl, ValidatorFn } from '@angular/forms';
    import { Customer } from './customer';

    function ratingRange(min: number, max: number): ValidatorFn {
      return (c: AbstractControl): { [key: string]: boolean } | null => {
        if (c.value !== null && (!isNaN(c.value) || c.value < min || c.value > max)) {
          return { 'range': true };
        }
        return null;
      };
    }

    @Component({
      selector: 'app-customer',
      templateUrl: './customer.component.html',
      styleUrls: ['./customer.component.css']
    })
    export class CustomerComponent implements OnInit {
      customerForm: FormGroup;
      customer = new Customer();

      constructor(private fb: FormBuilder) { }

      ngOnInit() {
        this.customerForm = this.fb.group({

```

updating the value of range in constructor

```

    11     return null;
    12   };
    13 }
    14
    15 @Component({
    16   selector: 'app-customer',
    17   templateUrl: './customer.component.html',
    18   styleUrls: ['./customer.component.css']
    19 })
    20 export class CustomerComponent implements OnInit {
    21   customerForm: FormGroup;
    22   customer = new Customer();
    23
    24   constructor(private fb: FormBuilder) { }
    25
    26   ngOnInit() {
    27     this.customerForm = this.fb.group({
    28       firstName: ['', [Validators.required, Validators.minLength(3)]],
    29       lastName: ['', [Validators.required, Validators.maxLength(50)]],
    30       email: ['', [Validators.required, Validators.email]],
    31       phone: '',
    32       notification: 'email',
    33       rating: [null, ratingRange(1,5)],
    34       sendCatalog: true
    35     });
    36   }
    37 }

```

## Cross Field Validation: Nested

Cross-field Validation: Nested FormGroup

```

this.customerForm = this.fb.group({
  firstName: ['', [Validators.required, Validators.minLength(3)]],
  lastName: ['', [Validators.required, Validators.maxLength(50)]],
  availability: this.fb.group({
    start: ['', Validators.required],
    end: ['', Validators.required]
  })
});

```

```

<div formGroupName="availability">
  ...
  <input formControlName="start"/>
  ...
  <input formControlName="end"/>
</div>

```

example;

The screenshot shows a web browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=5&mode=live>. The page title is "Angular Reactive Form". The main content is titled "Cross-field Validation" and shows a "Sign Up!" form. The "Email" and "Confirm Email" fields are grouped together and highlighted with a red border. The browser's sidebar on the right displays a "Table of Contents" for the course, listing various topics and their durations.

## Html Template for Confirm Email

The screenshot shows a Visual Studio Code editor window with the file "customer.component.html" open. The code is part of an Angular component and includes template-driven forms. A specific line of code for the "confirmEmail" field is highlighted with a red box. The browser's sidebar on the right displays a "Table of Contents" for the course, listing various topics and their durations.

Component updated by adding emailFormGroup

```

Activities Google Chrome ▾
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ Demo ▾ May 2 00:12 •
< > https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=5&mode=live
customer.component.ts - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
customer.component.html TS customer.component.ts
15 @Component({
16   selector: 'app-customer',
17   templateUrl: './customer.component.html',
18   styleUrls: ['./customer.component.css']
19 })
20 export class CustomerComponent implements OnInit {
21   customerForm: FormGroup;
22   customer = new Customer();
23
24   constructor(private fb: FormBuilder) { }
25
26   ngOnInit() {
27     this.customerForm = this.fb.group({
28       firstName: ['', [Validators.required, Validators.minLength(3)]],
29       lastName: ['', [Validators.required, Validators.maxLength(50)]],
30       emailGroup: this.fb.group([
31         email: ['', [Validators.required, Validators.email]],
32         confirmPassword: ['', Validators.required]
33       ]),
34       phone: '',
35       notification: 'email',
36       rating: [null, ratingRange(1, 5)],
37       sendCatalog: true
38     });
39
40   populateTestData(): void {
41   }
}
In 33, Col 10 Spaces: 2 UTF-8 LF TypeScript 3.0.3

```

Table of Contents Notes

- Setting Built-in Validation Rules 4m 45s
- Adjusting Validation Rules at ... 5m 27s
- Custom Validators 5m 35s
- Custom Validation with Paramet... 3m 6s
- Cross-Field Validation: Nested ... 4m 58s
- Cross-field Validation: Custom ... 5m 16s
- Checklists and Summary 4m 47s
- Reacting to Changes** 19m 57s
- Introduction 1m 4s
- Watching 4m 41s
- Reacting: Adjusting Validation ... 1m 48s
- Reacting: Displaying Validation... 5m 38s
- Reactive Transformations 3m 55s
- Checklists and Summary 2m 47s
- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context** 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Final Words

Since code has been moved to the new form groups html needs to be updated for both email and confirmPassword

```

Activities Google Chrome ▾
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ Demo ▾ May 2 00:14 •
< > https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m4&clip=5&mode=live
customer.component.html - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
customer.component.html TS customer.component.ts
51 </div>
52
53 <div formGroupName="emailGroup">
54   <div class="form-group row mb-2">
55     <label class="col-md-2 col-form-label"
56       for="emailId">Email</label>
57     <div class="col-md-8">
58       <input class="form-control"
59         id="emailId"
60         type="email"
61         placeholder="Email (required)"
62         formControlName="email"
63         [ngClass]="{'is-invalid': customerForm.get('emailGroup.email').touched ||
64           customerForm.get('emailGroup.email').dirty &&
65           !customerForm.get('emailGroup.email').valid }" />
66
67       <span class="invalid-feedback">
68         <span *ngIf="customerForm.get('emailGroup.email').errors?.required">
69           Please enter your email address.
70         </span>
71         <span *ngIf="customerForm.get('emailGroup.email').errors?.email">
72           Please enter a valid email address.
73         </span>
74       </div>
75     </div>
76
77   <div class="form-group row mb-2">...

```

Table of Contents Notes

- Setting Built-in Validation Rules 4m 43s
- Adjusting Validation Rules at ... 5m 27s
- Custom Validators 5m 35s
- Custom Validation with Paramet... 3m 6s
- Cross-Field Validation: Nested ... 4m 58s
- Cross-field Validation: Custom ... 5m 16s
- Checklists and Summary 4m 47s
- Reacting to Changes** 19m 57s
- Introduction 1m 4s
- Watching 4m 41s
- Reacting: Adjusting Validation ... 1m 48s
- Reacting: Displaying Validation... 5m 38s
- Reactive Transformations 3m 55s
- Checklists and Summary 2m 47s
- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context** 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Final Words

```

    <div class="form-group row mb-2">
      <label class="col-md-2 col-form-label" for="confirmEmailId">Confirm Email</label>
      <div class="col-md-8">
        <input class="form-control" id="confirmEmailId" type="email" placeholder="Confirm Email (required)" formControlName="confirmEmail" [ngClass]="{'is-invalid': (customerForm.get('emailGroup.confirmEmail').touched || customerForm.get('emailGroup.confirmEmail').dirty) && !customerForm.get('emailGroup.confirmEmail').valid }" >
        <span class="invalid-feedback">
          <span *ngIf="customerForm.get('emailGroup.confirmEmail').errors?.required">
            Please confirm your email address.
          </span>
        </span>
      </div>
    </div>
  </div>

  <div class="form-group row mb-2">
    <label class="col-md-2 col-form-label" for="phoneId">Phone</label>
    <div class="col-md-8">
      <input class="form-control" ...

```

In the above example validation message is not showing up

## Cross Field Validation:Custom Validator

### Cross-field Validation: Custom Validator

```

function dateCompare(c: AbstractControl): {[key: string]: boolean} | null {
  let startControl = c.get('start');
  let endControl = c.get('end');
  if (startControl.value !== endControl.value) {
    return { 'match': true };
  }
  return null;
}

this.customerForm = this.fb.group({
  firstName: ['', [Validators.required, Validators.minLength(3)]],
  lastName: ['', [Validators.required, Validators.maxLength(50)]],
  availability: this.fb.group({
    start: ['', Validators.required],
    end: ['', Validators.required]
  }, { validator: dateCompare })
});

```

This will add validation to the form group not the form controls.  
We have added a new method call emailMatchers.

```

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

    import { Customer } from './customer';

    function emailMatcher(c: AbstractControl): { [key: string]: boolean } | null {
      const emailControl = c.get('email');
      const confirmControl = c.get('confirmEmail');

      if (emailControl.pristine || confirmControl.pristine) {
        return null;
      }

      if (emailControl.value === confirmControl.value) {
        return null;
      }
      return { 'match': true };
    }

    function ratingRange(min: number, max: number): ValidatorFn {
      return (c: AbstractControl): { [key: string]: boolean } | null => {
        if (c.value !== null && (isNaN(c.value) || c.value < min || c.value > max)) {
          return { 'range': true };
        }
        return null;
      };
    }
  
```

We need to pass the object in the form group

```

    @Component({
      selector: 'app-customer',
      templateUrl: './customer.component.html',
      styleUrls: ['./customer.component.css']
    })
    export class CustomerComponent implements OnInit {
      customerForm: FormGroup;
      customer = new Customer();

      constructor(private fb: FormBuilder) { }

      ngOnInit() {
        this.customerForm = this.fb.group({
          firstName: ['', [Validators.required, Validators.minLength(3)]],
          lastName: ['', [Validators.required, Validators.maxLength(50)]],
          emailGroup: this.fb.group({
            email: ['', [Validators.required, Validators.email]],
            confirmPassword: ['', Validators.required],
            [validator: emailMatcher]
          }),
          phone: '',
          notification: 'email',
          rating: [null, ratingRange(1, 5)],
          sendCatalog: true
        });
      }

      populateTestData(): void {
        ...
      }
    }
  
```

New message is added with `customerForm.get("emailGroup").errors?.match`

```

customer.component.html
<div class="form-group row mb-2">
  <label class="col-md-2 col-form-label" for="confirmEmailId">Confirm Email</label>
  <div class="col-md-8">
    <input class="form-control" id="confirmEmailId" type="email" placeholder="Confirm Email (required)" formControlName="confirmEmail" [ngClass]="{'is-invalid': customerForm.get('emailGroup').errors || ((customerForm.get('emailGroup.confirmEmail').touched || customerForm.get('emailGroup.confirmEmail').dirty) && !customerForm.get('emailGroup.confirmEmail').valid) }" />
    <span class="invalid-feedback">
      <span *ngIf="customerForm.get('emailGroup.confirmEmail').errors?.required">
        Please confirm your email address.
      </span>
      <span *ngIf="customerForm.get('emailGroup').errors?.match">
        The confirmation does not match the email address.
      </span>
    </span>
  </div>
</div>
<div class="form-group row mb-2">

```

Table of Contents

- Setting Built-in Validation Rules 4m 45s
- Adjusting Validation Rules At... 5m 27s
- Custom Validators 5m 35s
- Custom Validation with Paramet... 3m 6s
- Cross-Field Validation: Nested ... 4m 58s
- Cross-field Validation: Custom ... 5m 16s**
- Checklists and Summary 4m 47s
- Reacting to Changes 19m 57s**
- Introduction 1m 4s
- Watching 4m 41s
- Reacting: Adjusting Validation ... 1m 48s
- Reacting: Displaying Validation... 5m 38s
- Reactive Transformations 3m 55s
- Checklists and Summary 2m 47s
- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s**
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Final Words

## Reacting to Changes

```

const phoneC
  (property) AbstractControl.valueChanges: Observable<any>
  A multicasting observable that emits an event every time the value of the control changes, in the UI or programmatically.
  phoneControl.valueChanges

```

Table of Contents

- Introduction 15m 28s**
- Template-driven vs. Reactive Forms 32m
- Building a Reactive Form 29m 36s
- Validation 35m 54s
- Reacting to Changes 18m 57s**
- Introduction 1m 4s
- Watching 4m 41s**
- Reacting: Adjusting Validation ... 1m 48s
- Reacting: Displaying Validation... 5m 38s
- Reactive Transformations 3m 55s
- Checklists and Summary 2m 47s
- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s**
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Final Words 5m 56s

Value change is better than state change. Example if have not touched the element we do not want to show the error.

The screenshot shows a Pluralsight video player interface. The main content area displays two snippets of TypeScript code:

```
const phoneControl = this.customerForm.get('phone');

phoneControl.valueChanges.subscribe();
```

```
const phoneControl = this.customerForm.get('phone');

phoneControl.statusChanges.subscribe();
```

To the right is a "Table of Contents" sidebar listing various video chapters:

- Introduction
- Template-driven vs. Reactive Forms
- Building a Reactive Form
- Validation
- Reacting to Changes
  - Introduction
  - Watching (highlighted)
  - Reacting: Adjusting Validation ...
  - Reacting: Displaying Validation...
  - Reactive Transformations
  - Checklists and Summary
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Final Words

Value change should happen at the ngOnInit but it must be added after the root Form Group otherwise it will be null. As shown below.

The screenshot shows a Visual Studio Code editor window. The left pane shows the file `customer.component.ts` with the following code:

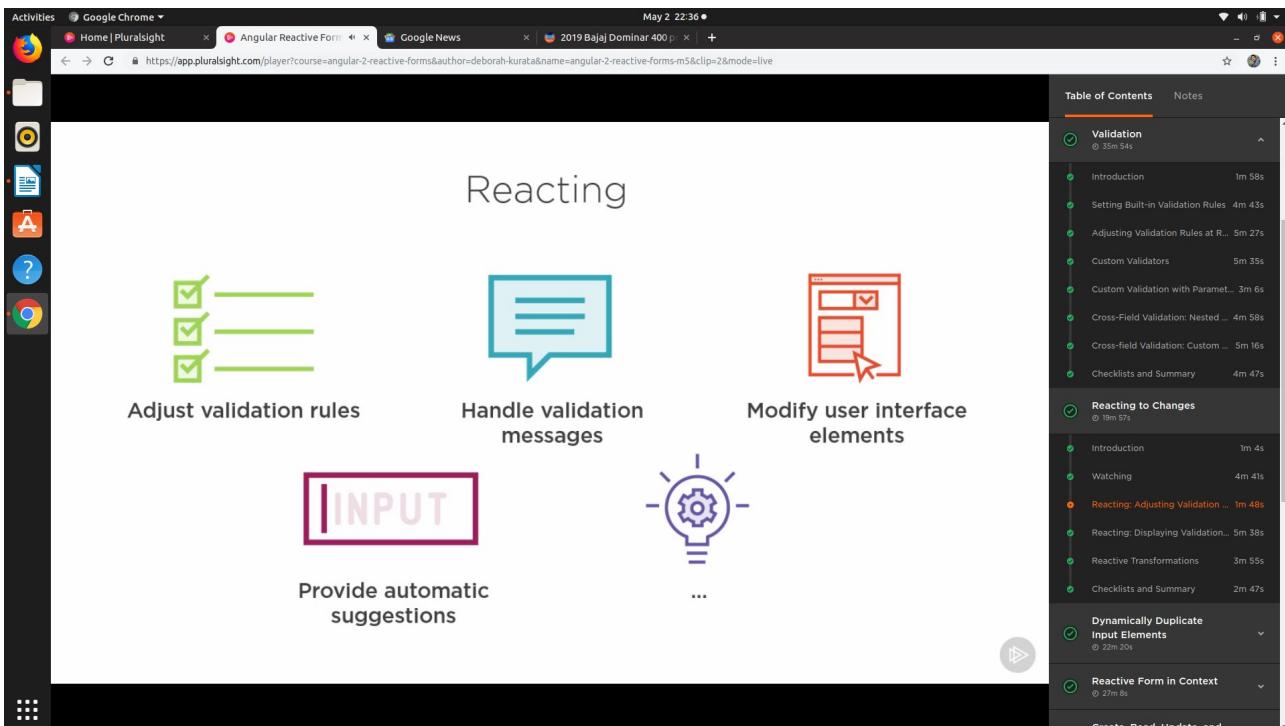
```
39
40  ngOnInit() {
41    this.customerForm = this.fb.group({
42      firstName: ['', [Validators.required, Validators.minLength(3)]],
43      lastName: ['', [Validators.required, Validators.maxLength(50)]],
44      emailGroup: this.fb.group({
45        email: ['', [Validators.required, Validators.email]],
46        confirmEmail: ['', Validators.required],
47      }, {validator: emailMatcher}),
48      phone: '',
49      notification: 'email',
50      rating: null,
51      sendCatalog: true
52    });

53    this.customerForm.get('notification').valueChanges.subscribe(
54      value => console.log(value)
55    );
56  }

57  populateTestData(): void {
58    this.customerForm.patchValue({
59      firstName: 'Jack',
60      lastName: 'Harkness',
61      sendCatalog: false
62    });
63  }
64}
65
```

A red arrow points to the line `this.customerForm = this.fb.group({`, indicating where the assignment should be placed. The right side of the screen shows the same "Table of Contents" as the previous screenshot.

## Implementation



With the watchers in place we can remove the click event from the notification and instead add a code in valueChange event inside ngOnInit.

```

customer.component.ts - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
TS customer.component.ts <-- customer.component.html

39
40 ngOnInit() {
41     this.customerForm = this.fb.group({
42         firstName: ['', [Validators.required, Validators.minLength(3)]],
43         lastName: ['', [Validators.required, Validators.maxLength(50)]],
44         emailGroup: this.fb.group({
45             email: ['', [Validators.required, Validators.email]],
46             confirmEmail: ['', Validators.required],
47             }, {validator: emailMatcher}),
48             phone: '',
49             notification: 'email',
50             rating: [null, ratingRange(1, 5)],
51             sendCatalog: true
52         });
53
54         this.customerForm.get('notification').valueChanges.subscribe(
55             | value => this.setNotification(value)
56         );
57
58
59 populateTestData(): void {
60     this.customerForm.patchValue({
61         firstName: 'Jack',
62         lastName: 'Harkness',
63         sendCatalog: false
64     });
65

```

The code in the ngOnInit() method is as follows:

```

ngOnInit() {
    this.customerForm = this.fb.group({
        firstName: ['', [Validators.required, Validators.minLength(3)]],
        lastName: ['', [Validators.required, Validators.maxLength(50)]],
        emailGroup: this.fb.group({
            email: ['', [Validators.required, Validators.email]],
            confirmEmail: ['', Validators.required],
            }, {validator: emailMatcher}),
            phone: '',
            notification: 'email',
            rating: [null, ratingRange(1, 5)],
            sendCatalog: true
        });
}

this.customerForm.get('notification').valueChanges.subscribe(
    | value => this.setNotification(value)
);

```

The line `this.customerForm.get('notification').valueChanges.subscribe(| value => this.setNotification(value))` is highlighted with a red box.

At the bottom of the code editor, the status bar shows: Line 55, Col 43, Spaces: 2, UTF-8, LF, TypeScript 3.0.3.

The screenshot shows a Visual Studio Code (VS Code) interface with a browser preview window. The browser displays a Pluralsight course page for "Angular Reactive Forms". The code editor shows `customer.component.ts` and `customer.component.html`. The `customer.component.html` file contains the following code:

```
<div class="form-group row mb-2">
  <label class="col-md-2 col-form-label pt-0">Send Notifications</label>
  <div class="col-md-8">
    <div class="form-check form-check-inline">
      <label class="form-check-label">
        <input class="form-check-input" type="radio" value="email" formControlName="notification"> Email
      </label>
    </div>
    <div class="form-check form-check-inline">
      <label class="form-check-label">
        <input class="form-check-input" type="radio" value="text" formControlName="notification"> Text
      </label>
    </div>
  </div>
<div class="form-group row mb-2">
  <label class="col-md-2 col-form-label">Rating</label>
  <div class="col-md-8">
```

The browser preview shows a form with two radio buttons labeled "Email" and "Text" under the heading "Send Notifications". Below the radio buttons is a label "Rating" followed by a blank input field.

### Moving Error Messages to the component class:

Create a datastructure with messages here validMessages and also a variable emailMessage.

## Step1:

The screenshot shows a browser window with a video player and a code editor. The video player is playing a course titled 'Angular Reactive Forms' by Deborah Kurata. The code editor in the foreground displays the 'customer.component.ts' file from the 'customer' component.

```
customer.component.ts - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help

TS customer.component.ts (customer.component.html)

34 export class CustomerComponent implements OnInit {
35   customerForm: FormGroup;
36   customer = new Customer();
37   emailMessage: string;
38
39   private validationMessages = {
40     required: 'Please enter your email address.',
41     email: 'Please enter a valid email address.'
42   };
43
44   constructor(private fb: FormBuilder) { }
45
46   ngOnInit() {
47     this.customerForm = this.fb.group({
48       firstName: ['', [Validators.required, Validators.minLength(3)]],
49       lastName: ['', [Validators.required, Validators.maxLength(50)]],
50       emailGroup: this.fb.group({
51         email: ['', [Validators.required, Validators.email]],
52         confirmEmail: ['', Validators.required],
53       }, { validator: emailMatcher }),
54       phone: '',
55       notification: 'email',
56       rating: [null, ratingRange(1, 5)],
57       sendCatalog: true
58     });
59
60     this.customerForm.get('notification').valueChanges.subscribe()
61   }
62 }

Ln 38, Col 3  Spaces:2  UTF-8  LF  TypeScript 3.0.3  🎧  📱
```

Step2: Add a watcher in ngOnInit as shown below

```

    45
    46    ngOnInit() {
    47      this.customerForm = this.fb.group({
    48        firstName: ['', [Validators.required, Validators.minLength(3)]],
    49        lastName: ['', [Validators.required, Validators.maxLength(50)]],
    50        emailGroup: this.fb.group({
    51          email: ['', [Validators.required, Validators.email]],
    52          confirmPassword: ['', Validators.required],
    53        }, { validator: emailMatcher }),
    54        phone: '',
    55        notification: 'email',
    56        rating: [null, ratingRange(1, 5)],
    57        sendCatalog: true
    58      });

    59      this.customerForm.get('notification').valueChanges.subscribe(
    60        value => this.setNotification(value)
    61      );
    62
    63      const emailControl = this.customerForm.get('emailGroup.email');
    64      emailControl.valueChanges.subscribe(
    65        value => this.setMessage(emailControl)
    66      );
    67    }

    68    populateTestData(): void {
    69      this.customerForm.patchValue({

```

Ln 67, Col 7 Spaces: 2 UTF-8 LF TypeScript 3.0.3

Step 3: Add a method setMessage since the method does not return anything its a void type

```

    66      value => this.setMessage(emailControl)
    67    );
    68
    69    populateTestData(): void {
    70      this.customerForm.patchValue({
    71        firstName: 'Jack',
    72        lastName: 'Harkness',
    73        sendCatalog: false
    74      });
    75
    76    }
    77
    78    save() {
    79      console.log(this.customerForm);
    80      console.log('Saved: ' + JSON.stringify(this.customerForm.value));
    81    }
    82
    83    setMessage(c: AbstractControl): void {
    84      this.emailMessage = '';
    85      if ((c.touched || c.dirty) && c.errors) {
    86        this.emailMessage = Object.keys(c.errors).map(
    87          key => this.validationMessages[key]] join(' '))
    88      }
    89    }
    90
    91    setNotification(notifyVia: string): void {

```

Ln 90, Col 1 Spaces: 2 UTF-8 LF TypeScript 3.0.3

Step 4: Remove everything from the class section and add emailMessage and also in span tag add interpolation {{emailMessage}} to display the message

```

customer.component.ts
52 <div formGroupName="emailGroup">
53   <div class="form-group row mb-2">
54     <label class="col-md-2 col-form-label"
55       for="emailId">Email</label>
56     <div class="col-md-8">
57       <input class="form-control"
58         id="emailId"
59         type="email"
60         placeholder="Email (required)"
61         formControlName="email"
62         [ngClass]="{'is-invalid': emailMessage }" />
63       <span class="invalid-feedback">
64         | {{ emailMessage }}</span>
65     </div>
66   </div>
67
68   <div class="form-group row mb-2">
69     <label class="col-md-2 col-form-label"
70       for="confirmEmailId">Confirm Email</label>
71     <div class="col-md-8">
72       <input class="form-control"
73         id="confirmEmailId"
74         type="email"
75         placeholder="Confirm Email (required)"
76         formControlName="confirmEmail" />
77     </div>
78

```

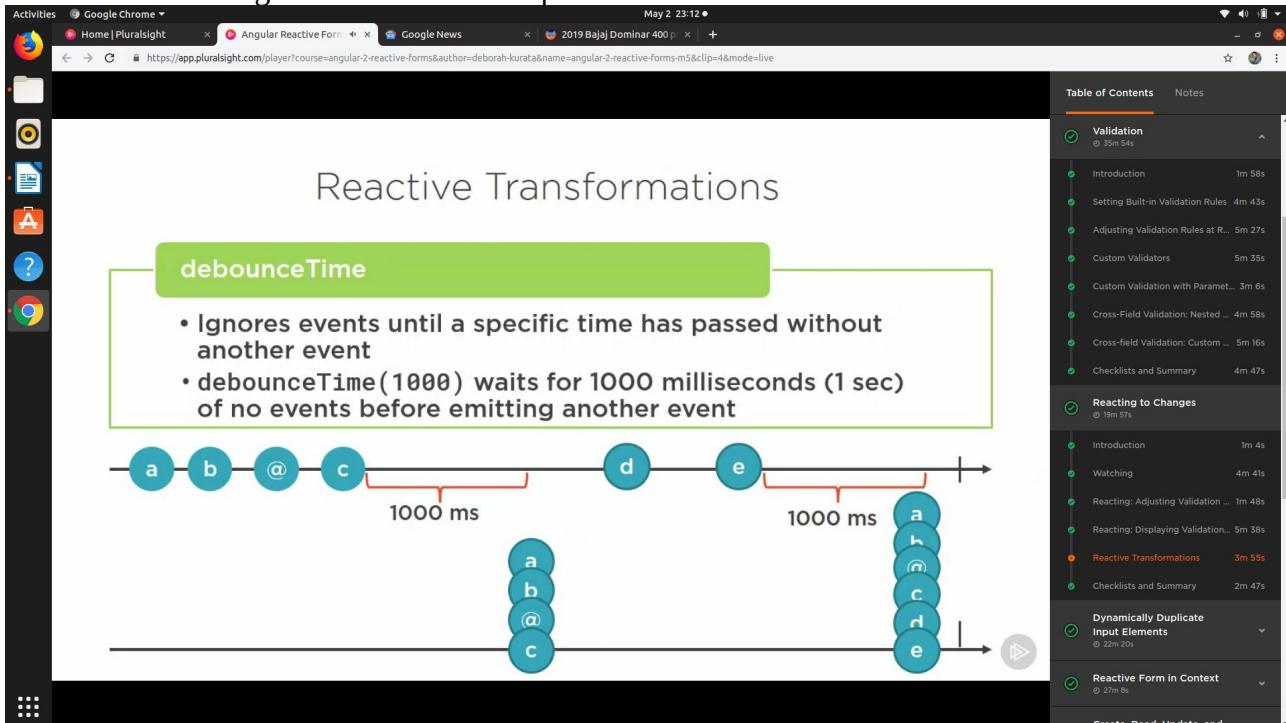
Ln 65, Col 31 Spaces: 2 UTF-8 LF HTML ⚡

Table of Contents Notes

- Validation** 35m 54s
  - Introduction 1m 58s
  - Setting Built-in Validation Rules 4m 43s
  - Adjusting Validation Rules At ... 5m 27s
  - Custom Validators 5m 35s
  - Custom Validation with Paramet... 3m 6s
  - Cross-Field Validation: Nested ... 4m 58s
  - Cross-field Validation: Custom ... 5m 16s
  - Checklists and Summary 4m 47s
- Reacting to Changes** 19m 57s
  - Introduction 1m 4s
  - Watching 4m 41s
  - Reacting: Adjusting Validation ... 1m 48s
  - Reacting: Displaying Validation... 5m 38s
  - Reactive Transformations 3m 55s
  - Checklists and Summary 2m 47s
- Dynamically Duplicate Input Elements** 22m 20s
- Reactive Form in Context** 27m 8s

Create, Read, Update, and

**Reactive Transformations:** In the above example there is one problem, when we type a single letter in the email input field it will start showing the error even before we havent typed the whole email which isn't right. To overcome this problem we use Reactive transformations.



**throttleTime** : This is useful when we receive way to many events movements as tracking mouse

**distinctUntilChange**: This is useful when tracking key events to prevent events when only controller shift keys are changed

Reactive Transformations

**throttleTime**

- Emits a value, then ignores subsequent values for a specific amount of time

**distinctUntilChanged**

- Suppresses duplicate consecutive items

Table of Contents

- Validation
  - Introduction
  - Setting Built-in Validation Rules
  - Adjusting Validation Rules at Runtime
  - Custom Validators
  - Custom Validation with Parameters
  - Cross-Field Validation: Nested Forms
  - Cross-Field Validation: Custom Validation
  - Checklists and Summary
- Reacting to Changes
  - Introduction
  - Watching
  - Reacting: Adjusting Validation Rules
  - Reacting: Displaying Validation Errors
  - Reactive Transformations
  - Checklists and Summary
- Dynamically Duplicate Input Elements
- Reactive Form in Context

Now adding reactive transformation to the existing code.

Add the dependency and then add the debounce time as shown below

```

1 const emailControl = this.customerForm.get('emailGroup.email');
2   emailControl.valueChanges.pipe(
3     debounceTime(1000)
4   ).subscribe(
5     value => this.setMessage(emailControl)
6   )
7 }

8 populateTestData(): void {
9   this.customerForm.patchValue({
10     firstName: 'Jack',
11     lastName: 'Harkness',
12     sendCatalog: false
13   });
14 }

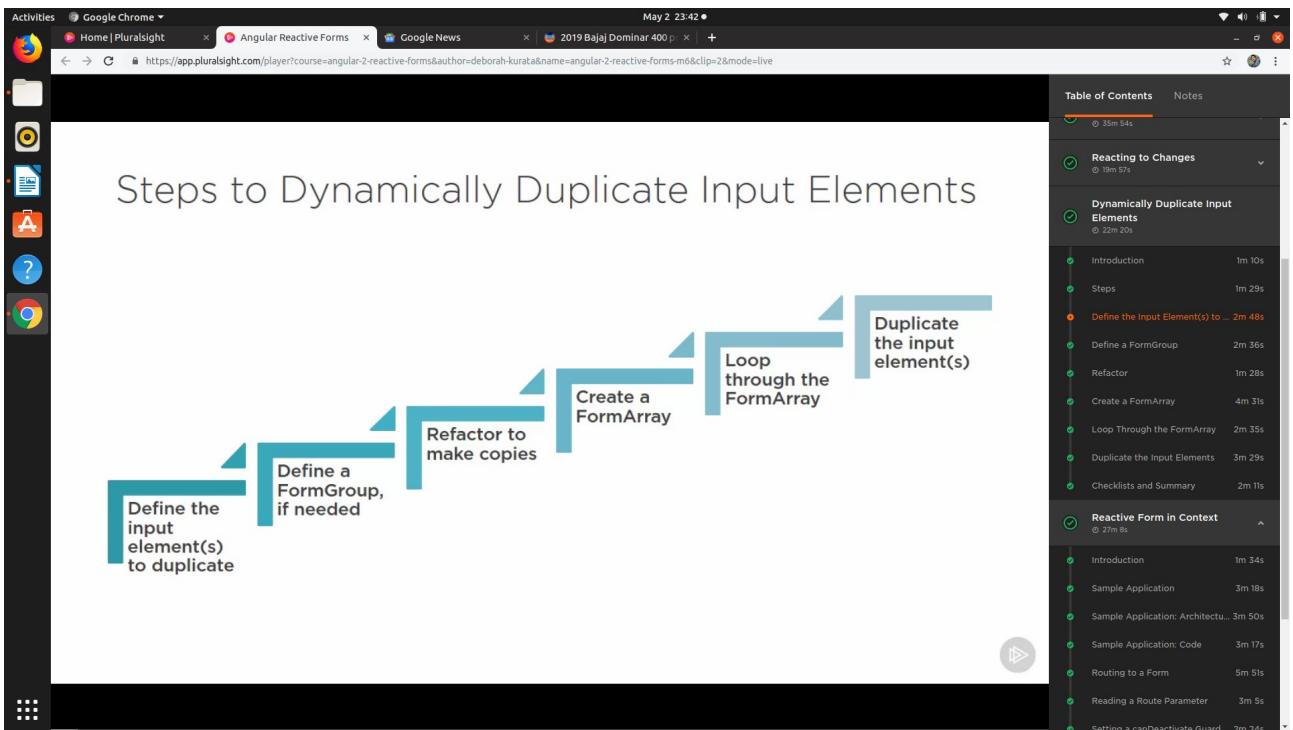
15 save() {
16   console.log(this.customerForm);
17   console.log('Saved: ' + JSON.stringify(this.customerForm.value));
18 }

19 setMessage(c: AbstractControl): void {
20 }

```

## Dynamically Duplicating Input Element

Steps to dynamically duplicate Input elements



## Add the fields in the form group

```

47
48     ngOnInit() {
49         this.customerForm = this.fb.group({
50             firstName: ['', [Validators.required, Validators.minLength(3)]],
51             lastName: ['', [Validators.required, Validators.maxLength(50)]],
52             emailGroup: this.fb.group({
53                 email: ['', [Validators.required, Validators.email]],
54                 confirmEmail: ['', Validators.required],
55             }, { validator: emailMatcher }),
56             phone: '',
57             notification: 'email',
58             rating: [null, ratingRange(1, 5)],
59             sendCatalog: true,
60             addressType: 'home',
61             street1: '',
62             street2: '',
63             city: '',
64             state: '',
65             zip: ''
66         });
67
68         this.customerForm.get('notification').valueChanges.subscribe(
69             value => this.setNotification(value)
70         );
71
72         const emailControl = this.customerForm.get('emailGroup.email');
73         emailControl.valueChanges.pipe(

```

The above name should match the fields in the html

```

171     <div *ngIf="customerForm.get('sendCatalog').value">
172       <div class="form-group row mb-2">
173         <label class="col-md-2 col-form-label pt-0">Address Type</label>
174         <div class="col-md-8">
175           <div class="form-check form-check-inline">
176             <label class="form-check-label">
177               <input class="form-check-input"
178                 id="addressTypeId"
179                 type="radio"
180                 value="home"
181                 formControlName="addressType"> Home
182             </label>
183           </div>
184           <div class="form-check form-check-inline">
185             <label class="form-check-label">
186               <input class="form-check-input"
187                 id="addressTypeId"
188                 type="radio"
189                 value="work"
190                 formControlName="addressType"> Work
191             </label>
192           </div>
193           <div class="form-check form-check-inline">
194             <label class="form-check-label">
195               <input class="form-check-input"
196                 id="addressTypeId"
197                 type="radio"
198                 value="other"
199                 formControlName="addressType"> Other
200           </label>
201         </div>
202       </div>
203     </div>

```

We need a form group for these new fields as per the second step in the steps diagram discussed earlier

**FormGroup**

**FormGroup**

**FormControl**    **FormControl**

**FormGroup**

**FormControl**    **FormControl**

**FormGroup**

**FormControl**

**FormGroup**

**FormControl**    **FormControl**

The screenshot shows a Pluralsight video player interface. The main content area displays five cards with the following text:

- Match the value of the form model to the data model
- Check touched, dirty, and valid state
- Watch for changes and react
- Perform cross field validation
- Dynamically duplicate the group

To the right of the content is a "Table of Contents" sidebar listing various video topics and their durations.

Topic	Duration
Define the Input Element(s) to ...	2m 48s
Define a FormGroup	2m 36s
Refactor	1m 28s
Create a FormArray	4m 31s
Loop Through the FormArray	2m 35s
Duplicate the Input Elements	3m 29s
Checklists and Summary	2m 11s
<b>Reactive Form in Context</b>	27m 8s
Introduction	1m 34s
Sample Application	3m 18s
Sample Application: Architectu...	3m 50s
Sample Application: Code	3m 17s
Routing to a Form	5m 51s
Reading a Route Parameter	3m 5s
Setting a canDeactivate Guard	2m 24s
Refactoring to a Custom Validat...	2m 28s
Checklists and Summary	1m 18s
<b>Create, Read, Update, and Delete (CRUD) Using HTTP</b>	44m 8s
<b>Final Words</b>	5m 56s

The screenshot shows a Visual Studio Code editor window with the file "customer.component.ts" open. The code defines a customer form with various fields and validation logic. A specific section of the code, which moves the form groups code inside a div tag, is highlighted with a red box.

```
50 firstName: ['', [Validators.required, Validators.minLength(3)]],  
51 lastName: ['', [Validators.required, Validators.maxLength(50)]],  
52 emailGroup: this.fb.group({  
53   email: ['', [Validators.required, Validators.email]],  
54   confirmEmail: ['', Validators.required],  
55   }, { validator: emailMatcher }),  
56 phone: '',  
57 notification: 'email',  
58 rating: [null, ratingRange(1, 5)],  
59 sendCatalog: true,  
60 addresses: this.fb.group({  
61   addressType: 'home',  
62   street1: '',  
63   street2: '',  
64   city: '',  
65   state: '',  
66   zip: ''  
67 })  
68 };  
69  
70 this.customerForm.get('notification').valueChanges.subscribe(  
71   value => this.setNotification(value)  
72 );  
73  
74 const emailControl = this.customerForm.get('emailGroup.email');  
75 emailControl.valueChanges.pipe(  
76   ...
```

Html Update: Moved the form groups code inside a div tag and added the new form group to it

The screenshot shows a developer environment with a code editor and a video player.

**Code Editor:**

- File:** customer.component.ts
- Content:** The code is a template-driven form (HTML) for a customer. It includes sections for addresses, phone numbers, and email addresses, each with its own form group and validation. A large button at the bottom handles the submission of the entire form.

```
<div *ngIf="customerForm.get('sendCatalog').value">
  <div formGroupName="addresses">
    <div class="form-group row mb-2">...
    </div>

    <div class="form-group row mb-2">...
    </div>
  </div>

  <div class="form-group row mb-2">...
  </div>

  <div class="form-group row mb-2">...
  </div>

  <div class="form-group row mb-2">...
  </div>

<div class="form-group row mb-2">
  <div class="offset-md-2 col-md-4">
    <button class="btn btn-primary mr-3"
           type="submit"
           style="width:80px"
           [title]="customerForm.valid ? 'Save your entered data' : 'Disabled until the form data is valid'"
           [disabled]="!customerForm.valid">
      Save
    </button>
    <button class="btn btn-outline-secondary" ...
  </div>
</div>
```

**Video Player:**

- Table of Contents:**
  - Define the Input Element(s) to ... 2m 48s
  - Define a FormGroup 2m 36s
  - Refactor 1m 28s
  - Create a FormArray 4m 31s
  - Loop Through the FormArray 2m 35s
  - Duplicate the Input Elements 3m 29s
  - Checklists and Summary 2m 11s
- Reactive Form in Context:**
  - Introduction 1m 34s
  - Sample Application 3m 18s
  - Sample Application: Architectu... 3m 50s
  - Sample Application: Code 3m 17s
  - Routing to a Form 5m 51s
  - Reading a Route Parameter 3m 5s
  - Setting a canDeactivate Guard 2m 24s
  - Refactoring to a Custom Validat... 2m 28s
  - Checklists and Summary 1m 18s
- Create, Read, Update, and Delete (CRUD) Using HTTP:**
  - Final Words 5m 56s

## Refactoring to make copies:

Activities Google Chrome May 3 00:20

Home | Pluralsight Angular Reactive Form https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m6&clip=4&mode=live

## Creating a FormGroup in a Method

```
buildAddress(): FormGroup {
  return this.fb.group({
    addressType: 'home',
    street1: '',
    street2: '',
    city: '',
    state: '',
    zip: ''
  });
}

this.customerForm = this.fb.group({
  ...
  addresses: this.buildAddress()
});
```

Table of Contents Notes

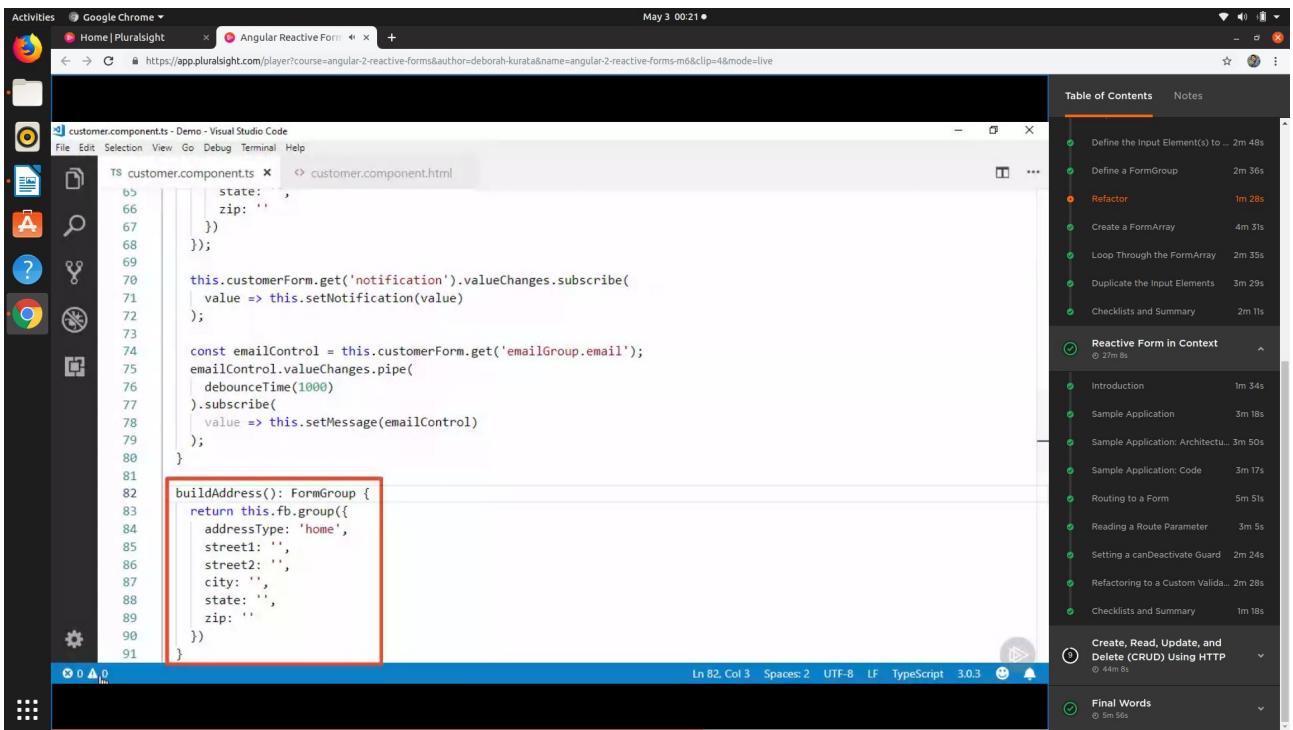
- Define the Input Element(s) to ... 2m 48s
- Define a FormGroup 2m 36s
- Refactor 1m 28s
- Create a FormArray 4m 31s
- Loop Through the FormArray 2m 35s
- Duplicate the Input Elements 3m 29s
- Checklists and Summary 2m 11s

Reactive Form in Context

- Introduction 1m 34s
- Sample Application 3m 18s
- Sample Application: Architectu... 3m 50s
- Sample Application: Code 3m 17s
- Routing to a Form 5m 51s
- Reading a Route Parameter 3m 5s
- Setting a canDeactivate Guard 2m 24s
- Refactoring to a Custom Validat... 2m 28s
- Checklists and Summary 1m 18s

Create, Read, Update, and Delete (CRUD) Using HTTP

Final Words



```

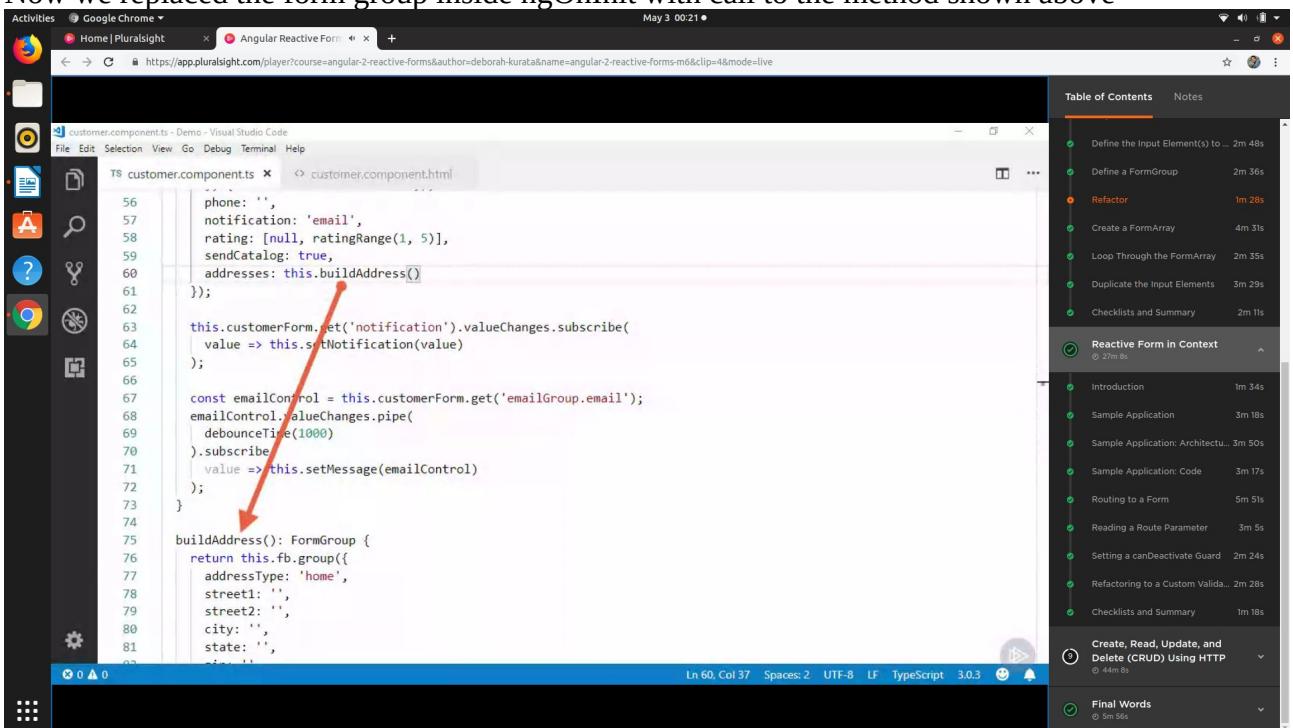
customer.component.ts - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
TS customer.component.ts x customer.component.html
55   state: '',
56   zip: ''
57 });
58 });
59 );
60 const emailControl = this.customerForm.get('emailGroup.email');
61 emailControl.valueChanges.pipe(
62   debounceTime(1000)
63 ).subscribe(
64   value => this.setMessage(emailControl)
65 );
66
67 buildAddress(): FormGroup {
68   return this.fb.group({
69     addressType: 'home',
70     street1: '',
71     street2: '',
72     city: '',
73     state: '',
74     zip: ''
75   })
76 }

```

Table of Contents Notes

- Define the Input Element(s) to ... 2m 48s
- Define a FormGroup 2m 36s
- Refactor** 1m 28s
- Create a FormArray 4m 31s
- Loop Through the FormArray 2m 35s
- Duplicate the Input Elements 3m 29s
- Checklists and Summary 2m 11s
- Reactive Form in Context** 27m 8s
- Introduction 1m 34s
- Sample Application 3m 18s
- Sample Application: Architecture 3m 50s
- Sample Application: Code 3m 17s
- Routing to a Form 5m 51s
- Reading a Route Parameter 3m 5s
- Setting a canDeactivate Guard 2m 24s
- Refactoring to a Custom Validator 2m 28s
- Checklists and Summary 1m 18s
- Create, Read, Update, and Delete (CRUD) Using HTTP** 44m 8s
- Final Words 5m 56s

Now we replaced the form group inside ngOnInit with call to the method shown above



```

customer.component.ts - Demo - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
TS customer.component.ts x customer.component.html
56   phone: '',
57   notification: 'email',
58   rating: [null, ratingRange(1, 5)],
59   sendCatalog: true,
60   addresses: this.buildAddress()
61 };
62
63 this.customerForm.get('notification').valueChanges.subscribe(
64   value => this.setNotification(value)
65 );
66
67 const emailControl = this.customerForm.get('emailGroup.email');
68 emailControl.valueChanges.pipe(
69   debounceTime(1000)
70 ).subscribe(
71   value => this.setMessage(emailControl)
72 );
73
74 buildAddress(): FormGroup {
75   return this.fb.group({
76     addressType: 'home',
77     street1: '',
78     street2: '',
79     city: '',
80     state: '',
81     zip: ''
82   })
83 }

```

Table of Contents Notes

- Define the Input Element(s) to ... 2m 48s
- Define a FormGroup 2m 36s
- Refactor** 1m 28s
- Create a FormArray 4m 31s
- Loop Through the FormArray 2m 35s
- Duplicate the Input Elements 3m 29s
- Checklists and Summary 2m 11s
- Reactive Form in Context** 27m 8s
- Introduction 1m 34s
- Sample Application 3m 18s
- Sample Application: Architecture 3m 50s
- Sample Application: Code 3m 17s
- Routing to a Form 5m 51s
- Reading a Route Parameter 3m 5s
- Setting a canDeactivate Guard 2m 24s
- Refactoring to a Custom Validator 2m 28s
- Checklists and Summary 1m 18s
- Create, Read, Update, and Delete (CRUD) Using HTTP** 44m 8s
- Final Words 5m 56s

**Creating Form Array:** In form array fields can be accessed as index just like a normal array. It can have multiple form control groups in it.

The diagram illustrates two methods for creating a FormArray. On the left, under 'FormArray', there are three examples. The first example shows a single FormArray containing two FormControl elements. The second example shows a FormArray containing a single FormGroup, which itself contains two FormControl elements. The third example shows a FormArray containing two FormGroup elements, each with two FormControl elements. On the right, under 'Create a FormArray', there are three numbered steps: Step 0 shows a FormArray with index 0 containing a FormGroup; Step 1 shows the same FormArray with index 1 containing another FormGroup; Step 2 shows the same FormArray with index 2 containing yet another FormGroup.

These are the two ways of creating a form array. As we are familiar with the form builder approach we will use it instead.

The screenshot shows two code snippets for creating a FormArray. The top snippet is:

```
this.myArray = new FormArray([...]);
```

The bottom snippet is:

```
this.myArray = this.fb.array([...]);
```

Form Array as address is created below with method call to build address so that array is initialized when first time form is called.

```
ngOnInit() {
    this.customerForm = this.fb.group({
        firstName: ['', [Validators.required, Validators.minLength(3)]],
        lastName: ['', [Validators.required, Validators.maxLength(50)]],
        emailGroup: this.fb.group({
            email: ['', [Validators.required, Validators.email]],
            confirmPassword: ['', Validators.required],
        }, { validator: emailMatcher }),
        phone: '',
        notification: 'email',
        rating: [null, ratingRange(1, 5)],
        sendCatalog: true,
        addresses: this.fb.array([ this.buildAddress() ])
    });

    this.customerForm.get('notification').valueChanges.subscribe(
        value => this.setNotification(value)
    );

    const emailControl = this.customerForm.get('emailGroup.email');
    emailControl.valueChanges.pipe(
        debounceTime(1000)
    ).subscribe(
        value => this.setMessage(emailControl)
    );
}
```

```
templateUrl: './customer.component.html',
styleUrls: ['./customer.component.css']
})
export class CustomerComponent implements OnInit {
    customerForm: FormGroup;
    customer = new Customer();
    emailMessage: string;

    get addresses(): FormArray{
        return <FormArray>this.customerForm.get('addresses');
    }

    private validationMessages = {
        required: 'Please enter your email address.',
        email: 'Please enter a valid email address.'
    };

    constructor(private fb: FormBuilder) { }

    ngOnInit() {
        this.customerForm = this.fb.group({
            firstName: ['', [Validators.required, Validators.minLength(3)]],
            lastName: ['', [Validators.required, Validators.maxLength(50)]],
            emailGroup: this.fb.group({
                email: ['', [Validators.required, Validators.email]],
                confirmPassword: ['', Validators.required],
            }, { validator: emailMatcher })
        });
    }
}
```

Now we put our html code inside a new div and gave it the name of the form array

```

    <div *ngIf="customerForm.get('sendCatalog').value">
      <div formArrayName="addresses">
        <div formGroupName="addresses">...
          <div class="form-group row mb-2">...
            ...
          </div>
          <div class="form-group row mb-2">...
            ...
          </div>
          <div class="form-group row mb-2">...
            ...
          </div>
        </div>
      </div>
      <div class="form-group row mb-2">
        <div class="offset-md-2 col-md-4">
          <button class="btn btn-primary mr-3"
            type="submit"
            style="width:80px"
            [title]="customerForm.valid ? 'Save your entered data' : 'Disabled until the form data is valid'"
            [disabled]="!customerForm.valid">
            Save
        </button>
      </div>
    </div>
  
```

Now we will make our form group inside the div as value 0, as in the component code we are calling a buildAddress method as it will create 1 object of the form group when called first time.

```

    <div *ngIf="customerForm.get('sendCatalog').value">
      <div formArrayName="addresses">
        <div formGroupName="0">
          <div class="form-group row mb-2">...
            ...
          </div>
          <div class="form-group row mb-2">...
            ...
          </div>
          <div class="form-group row mb-2">...
            ...
          </div>
        </div>
      </div>
      <div class="form-group row mb-2">
        <div class="offset-md-2 col-md-4">
          <button class="btn btn-primary mr-3"
            type="submit"
            style="width:80px"
            [title]="customerForm.valid ? 'Save your entered data' : 'Disabled until the form data is valid'"
            [disabled]="!customerForm.valid">
            Save
        </button>
      </div>
    </div>
  
```

## Looping through the Form Array:

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular Reactive Forms ▾ May 3 00:34 ●

Angular Reactive Forms

Dynamically Duplicate Input Elements : Loop Through the FormArray

## Looping Through a FormArray

```
<div formArrayName="addresses"
      *ngFor="let address of addresses.controls; let i=index">

    <div [formGroupName]=“i”>
      ...
    </div>

</div>
```

Table of Contents Notes

- Define the Input Element(s) to ... 2m 48s
- Define a FormGroup 2m 36s
- Refactor 1m 28s
- Create a FormArray 4m 31s
- Loop Through the FormArray 2m 35s
- Duplicate the Input Elements 3m 29s
- Checklists and Summary 2m 11s

Reactive Form in Context 0 27m 8s

- Introduction 1m 34s
- Sample Application 3m 18s
- Sample Application: Architectu... 3m 50s
- Sample Application: Code 3m 17s
- Routing to a Form 5m 51s
- Reading a Route Parameter 3m 5s
- Setting a canActivate Guard 2m 24s
- Refactoring to a Custom Validat... 2m 28s
- Checklists and Summary 1m 18s

Create, Read, Update, and Delete (CRUD) Using HTTP 0 44m 8s

Final Words 0 5m 58s

for in the individual fields inside the array will be replaced by attr.for field and the id of the individual field will be updated as shown below

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular Reactive Forms ▾ May 3 00:36 ●

customer.component.html - Demo - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

```
171   <div *ngIf="customerForm.get('sendCatalog').value">
172     <div formArrayName="addresses"
173       *ngFor="let address of addresses.controls; let i=index">
174         <div [formGroupName]=“i”>
175           <div class="form-group row mb-2">...
176             </div>
177
178           <div class="form-group row mb-2">
179             <label class="col-md-2 col-form-label"
180               attr.for="{{ 'street1Id' + i }}>Street Address 1</label>
181             <div class="col-md-8">
182               <input class="form-control"
183                 id="{{ 'street1Id' + i }}"
184                 type="text"
185                 placeholder="Street address"
186                 formControlName="street1">
187             </div>
188           </div>
189
190           <div class="form-group row mb-2">...
191         </div>
192
193           <div class="form-group row mb-2">...
194         </div>
195       </div>
196     </div>
197   </div>
```

Table of Contents Notes

- Define the Input Element(s) to ... 2m 48s
- Define a FormGroup 2m 36s
- Refactor 1m 28s
- Create a FormArray 4m 31s
- Loop Through the FormArray 2m 35s
- Duplicate the Input Elements 3m 29s
- Checklists and Summary 2m 11s

Reactive Form in Context 0 27m 8s

- Introduction 1m 34s
- Sample Application 3m 18s
- Sample Application: Architectu... 3m 50s
- Sample Application: Code 3m 17s
- Routing to a Form 5m 51s
- Reading a Route Parameter 3m 5s
- Setting a canActivate Guard 2m 24s
- Refactoring to a Custom Validat... 2m 28s
- Checklists and Summary 1m 18s

Create, Read, Update, and Delete (CRUD) Using HTTP 0 44m 8s

Final Words 0 5m 58s

## Duplicate the Input Elements:

A screenshot of a Pluralsight video player interface. The video title is "Angular Reactive Forms" and the specific topic is "Dynamically Duplicate Input Elements : Duplicate the Input Elements". The video player shows a code snippet in a text editor and a corresponding UI component. A red arrow points from the code snippet to the UI component. The UI component is a button labeled "Add Another Address". The video player has a "Table of Contents" sidebar on the right.

```
addAddress(): void {
  this.addresses.push(this.buildAddress());
}
```

```
<button class="btn btn-primary"
        type="button"
        (click)="addAddress()">
  Add Another Address
</button>
```

Add a new method in the component

A screenshot of a code editor showing the file "customer.component.ts". A new method "addAddress()" has been added to the component. The code editor has a "Table of Contents" sidebar on the right.

```
addAddress(): void {
  this.addresses.push(this.buildAddress());
}
```

Adding a button for creating a multi value

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular Reactive Forms ▾ https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m6&clip=7&mode=live

### Angular Reactive Forms

Dynamically Duplicate Input Elements : Duplicate the Input Elements

customer.component.html - Demo - Visual Studio Code

```

<div class="form-group row mb-2">
  <div class="col-md-4">
    <button class="btn btn-outline-primary"
      type="button"
      [disabled]="addresses.valid"
      (click)="addAddress()>
      Add Another Address
    </button>
  </div>
</div>

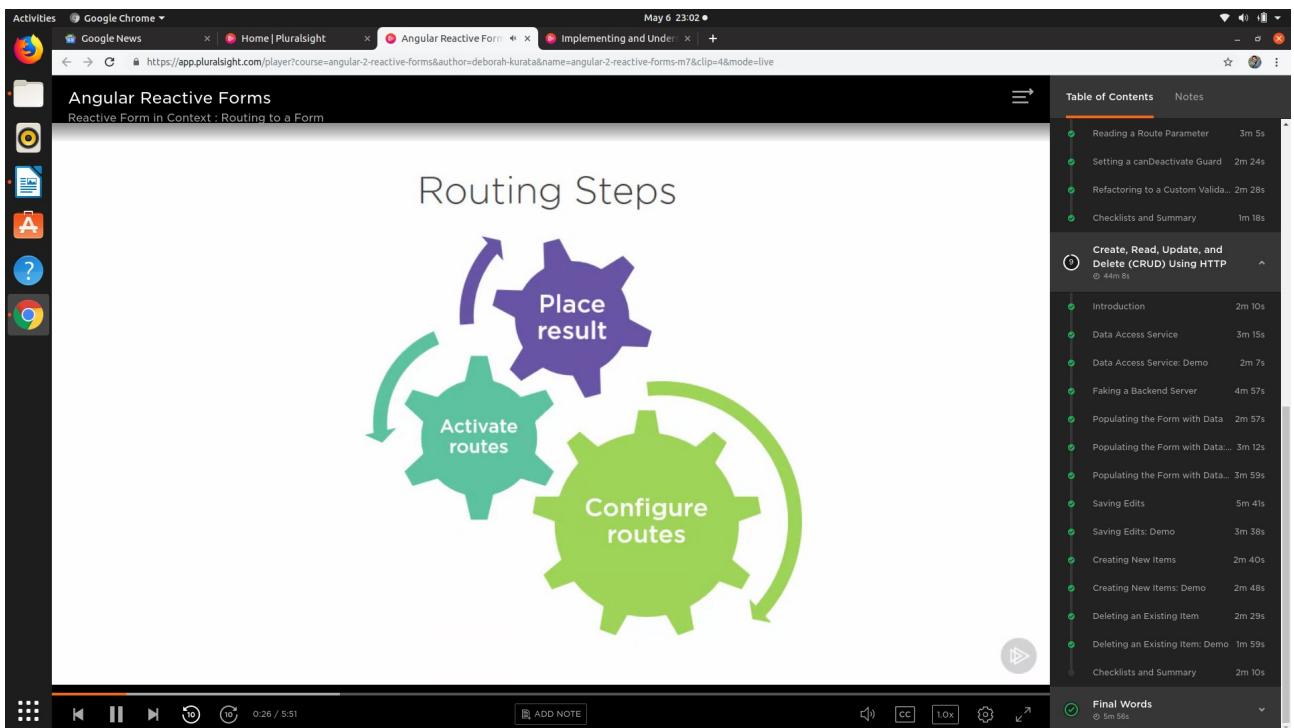
```

Table of Contents Notes

- Define the Input Element(s) to ... 2m 48s
- Define a FormGroup 2m 36s
- Refactor 1m 28s
- Create a FormArray 4m 31s
- Loop Through the FormArray 2m 35s
- Duplicate the Input Elements** 3m 29s
- Checklists and Summary 2m 11s
- Reactive Form In Context** 27m 8s
- Introduction 1m 34s
- Sample Application 3m 18s
- Sample Application: Architecture 3m 50s
- Sample Application: Code 3m 17s
- Routing to a Form 5m 51s
- Reading a Route Parameter 3m 5s
- Setting a canActivate Guard 2m 24s
- Refactoring to a Custom Validator 2m 28s
- Checklists and Summary 1m 18s
- Create, Read, Update, and Delete (CRUD) Using HTTP** 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data... 2m 57s
- Populating the Form with Data... 3m 12s
- Populating the Form with Data... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words** 5m 58s

Ln 281, Col 13 Spaces: 2 UTF-8 LF HTML ⚡ ADD NOTE 🔍 CC 1.0x ⚡ ↻

## Reactive Form In Context



The screenshot shows a video player interface with a dark theme. The main content area displays the following code:

```
[  
  { path: 'products', component: ProductListComponent },  
  { path: 'products/:id', component: ProductDetailComponent },  
  { path: 'products/:id/edit', component: ProductEditComponent }  
]
```

To the right of the video player is a sidebar titled "Table of Contents". The table of contents includes sections like "Validation", "Reacting to Changes", "Dynamically Duplicate Input Elements", "Reactive Form in Context", "Create, Read, Update, and Delete (CRUD) Using HTTP", and "Checklists and Summary". Each section has a thumbnail and a duration.

**canActivate:** This route checks the specific criteria before allowing navigation to particular route.  
**CanDeactivate:** It checks the specific criteria before allowing navigation away from particular route

The screenshot shows a video player interface with a dark theme. The main content area displays the following code, with the "canDeactivate" guard highlighted by a red box:

```
[  
  { path: 'products', component: ProductListComponent },  
  { path: 'products/:id',  
    canActivate: [ ProductDetailGuard ],  
    component: ProductDetailComponent },  
  { path: 'products/:id/edit',  
    canDeactivate: [ ProductEditGuard ],  
    component: ProductEditComponent }  
]
```

To the right of the video player is a sidebar titled "Table of Contents". The table of contents includes sections like "Validation", "Reacting to Changes", "Dynamically Duplicate Input Elements", "Reactive Form in Context", "Create, Read, Update, and Delete (CRUD) Using HTTP", and "Checklists and Summary". Each section has a thumbnail and a duration.

**Example:**

```

15
16 @NgModule({
17   imports: [
18     SharedModule,
19     ReactiveFormsModule,
20     InMemoryWebApiModule.forRoot(ProductData),
21     RouterModule.forChild([
22       { path: 'products', component: ProductListComponent },
23       { path: 'products/:id', component: ProductDetailComponent },
24       {
25         path: 'products/:id/edit',
26         canActivate: [ProductEditGuard],
27         component: ProductEditComponent
28       }
29     ]),
30   ],
31   declarations: [
32     ProductListComponent,
33     ProductDetailComponent,
34     ProductEditComponent
35   ]
36 })
37 export class ProductModule { }
38

```

Table of Contents

- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context**
- Introduction
- Sample Application
- Sample Application: Architecture
- Sample Application: Code
- Routing to a Form**
- Reading a Route Parameter
- Setting a canDeactivate Guard
- Refactoring to a Custom Validator
- Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Introduction
- Data Access Service
- Data Access Service: Demo

Tying route to action using router link directive and use property binding to map router link to an array.

Here we pass 0 to the product edit route, here in this scenario 0 means new product request

```

app.component.ts
...
@Component({
  selector: 'pm-root',
  template:
    <ul class='navbar-nav'>
      <li><a [routerLink]="/welcome">Home</a></li>
      <li><a [routerLink]="/products">Product List</a></li>
      <li><a [routerLink]="/products', '0', 'edit'">
        Add Product</a>
      </li>
    </ul>
})

```

Table of Contents

- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context**
- Introduction
- Sample Application
- Sample Application: Architecture
- Sample Application: Code
- Routing to a Form**
- Reading a Route Parameter
- Setting a canDeactivate Guard
- Refactoring to a Custom Validator
- Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Introduction
- Data Access Service
- Data Access Service: Demo

Angular Reactive Forms

Reactive Form in Context : Routing to a Form

File Edit Selection View Go Debug Terminal Help

TS app.component.ts x product-list.component.html product-detail.component.html

```

1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'pm-root',
5   template: `
6     <nav class='navbar navbar-expand navbar-light bg-light'>
7       <a class='navbar-brand'>{{pageTitle}}</a>
8       <ul class='navbar-nav'>
9         <li class='nav-item'><a class='nav-link' routerLinkActive='active' [routerLink]="/welcome">Home</a>
10        </li>
11        <li class='nav-item'><a class='nav-link' routerLinkActive='active' [routerLinkActiveOptions]={"exact: true"} [routerLink]="/products">Product List</a>
12        </li>
13        <li class='nav-item'><a class='nav-link' routerLinkActive='active' [routerLinkActiveOptions]={"exact: true"} [routerLink]="/products/0/edit">Add Product</a>
14        </li>
15      </ul>
16    </nav>
17    <div class='container'>
18      <router-outlet></router-outlet>
19    </div>
20  `,
21  styleUrls: ['./app.component.css']
22})
23 export class AppComponent {
24   pageTitle = 'Acme Product Management';
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

```

In 16, Col 38 Spaces: 2 UTF-8 CRLF TypeScript 3.0.3

Table of Contents Notes

- Validation (35m 54s)
- Reacting to Changes (19m 57s)
- Dynamically Duplicate Input Elements (22m 20s)
- Reactive Form in Context (27m 6s)
  - Introduction
  - Sample Application
  - Sample Application: Architecture
  - Sample Application: Code
  - Routing to a Form (5m 51s)
  - Reading a Route Parameter
  - Setting a canActivate Guard
  - Refactoring to a Custom Validator
  - Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP (44m 8s)
  - Introduction
  - Data Access Service
  - Data Access Service: Demo

## Product list component:

Activities Google Chrome

Google News Home Pluralsight Angular Reactive Form Implementing and Under

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m7&clip=4&mode=live

File Edit Selection View Go Debug Terminal Help

TS product-list.component.html - APM - Visual Studio Code

product-list.component.html x product-detail.component.html

```

44   <td>
45     <img [src]= "product.imageUrl"
46     [title]= "product.productName"
47     [style.width.px]= "imageWidth"
48     [style.margin.px]= "imageMargin">
49   </td>
50   <td>
51     <a [routerLink]=["/products", product.id]>
52       {{ product.productName }}
53     </a>
54   </td>
55   <td>{{ product.productCode }}</td>
56   <td>{{ product.releaseDate }}</td>
57   <td>{{ product.price | currency:"USD": "symbol": "1.2-2" }}</td>
58   <td>
59     <pm-star [rating]= "product.starRating">
60     </pm-star>
61   </td>
62   <td>
63     <button class="btn btn-outline-primary btn-sm"
64       [routerLink]=["/products", product.id, 'edit']>
65       Edit
66     </button>
67   </td>
68   </tr>
69 </tbody>
70 </table>

```

In 70, Col 11 Spaces: 2 UTF-8 CRLF HTML

Table of Contents Notes

- Validation (35m 54s)
- Reacting to Changes (19m 57s)
- Dynamically Duplicate Input Elements (22m 20s)
- Reactive Form in Context (27m 6s)
  - Introduction
  - Sample Application
  - Sample Application: Architecture
  - Sample Application: Code
  - Routing to a Form (5m 51s)
  - Reading a Route Parameter
  - Setting a canActivate Guard
  - Refactoring to a Custom Validator
  - Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP (44m 8s)
  - Introduction
  - Data Access Service
  - Data Access Service: Demo

## Product Detail:

The screenshot shows a browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m7&clip=4&mode=live>. The page displays the 'Angular Reactive Forms' course content. On the left, there are two tabs: 'product-list.component.html' and 'product-detail.component.html'. The right side shows a 'Table of Contents' with various sections and their durations. A red box highlights a specific line of code in the 'product-detail.component.html' tab:

```

    <button class="btn btn-outline-primary" style="width:80px;" [routerLink]="/products", product.id, 'edit'>
      Edit
    </button>
  
```

For Reading parameter from the routes we use ActivatedRoutes service.

It can be retrieved using two ways

1) **Snapshot:** Use this approach if you need to get only the initial value of the parameter

2) Using **observable:** If we need the parameter to change without changing the page then we should use the observable approach. It provides notifications when url parameter changes.

The screenshot shows a browser window with the URL <https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m7&clip=5&mode=live>. The page displays the 'Reading Parameters from a Route' section of the course. On the left, there is a code editor with two examples. The first example uses the 'snapshot' property of the ActivatedRoute service:

```

{ path: 'products/:id/edit', component: ProductEditComponent }

```

The second example uses the 'paramMap' observable of the ActivatedRoute service:

```

import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
  let id = +this.route.snapshot.paramMap.get('id');
  ...
}

constructor(private route: ActivatedRoute) {
  this.sub = this.route.paramMap.subscribe(
    params => {
      const id = +params.get('id');
      ...
    });
}

```

To ensure the subscription is appropriately cleaned up we need to call **ngOnDestroy** method to unsubscribe.

```
product-edit.component.ts
62 this.productForm = this.fb.group({
63   productName: ['', [Validators.required,
64     Validators.minLength(3),
65     Validators.maxLength(50)]],
66   productCode: ['', Validators.required],
67   starRating: ['', NumberValidators.range(1, 5)],
68   tags: this.fb.array([]),
69   description: ''
70 });
71
72 // Read the product Id from the route parameter
73 this.sub = this.route.paramMap.subscribe(
74   params => {
75     const id = +params.get('id');
76     this.getProduct(id);
77   }
78 );
79
80 }
81
82 ngOnDestroy(): void {
83   this.sub.unsubscribe();
84 }
85
86 ngAfterViewInit(): void {
87   // Watch for the blur event from any input element on the form.
88 }
```

Table of Contents

- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
  - Introduction
  - Sample Application
  - Sample Application: Architecture
  - Sample Application: Code
  - Routing to a Form
  - Reading a Route Parameter
  - Setting a canDeactivate Guard
  - Refactoring to a Custom Validator
  - Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP
  - Introduction
  - Data Access Service
  - Data Access Service: Demo

## Building a Guard:

### Building a Guard

```
product-edit.guard.ts
import { Injectable } from '@angular/core';
import { CanDeactivate } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
export class ProductEditGuard
  implements CanDeactivate<ProductEditComponent> {

  canDeactivate(component: ProductEditComponent): boolean {
    ...
  }
}
```

Table of Contents

- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
  - Introduction
  - Sample Application
  - Sample Application: Architecture
  - Sample Application: Code
  - Routing to a Form
  - Reading a Route Parameter
  - Setting a canDeactivate Guard
  - Refactoring to a Custom Validator
  - Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP
  - Introduction
  - Data Access Service
  - Data Access Service: Demo

```

1 import { Injectable } from '@angular/core';
2 import { CanDeactivate } from '@angular/router';
3 import { Observable } from 'rxjs';
4
5 import { ProductEditComponent } from './product-edit.component';
6
7 @Injectable({
8   providedIn: 'root',
9 })
10 export class ProductEditGuard implements CanDeactivate<ProductEditComponent> {
11   canDeactivate(component: ProductEditComponent): Observable<boolean | Promise<boolean> | boolean {
12     if (component.productForm.dirty) {
13       const productName = component.productForm.get('productName').value || 'New Product';
14       return confirm(`Navigate away and lose all changes to ${productName}?`);
15     }
16     return true;
17   }
18 }

```

Table of Contents

- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Introduction
- Sample Application
- Sample Application: Architecture
- Sample Application: Code
- Routing to a Form
- Reading a Route Parameter
- Setting a canDeactivate Guard
- Refactoring to a Custom Validator
- Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Introduction
- Data Access Service
- Data Access Service: Demo

## Now tying the route to the route

```

11 import { ProductListComponent } from './product-list.component';
12 import { ProductDetailComponent } from './product-detail.component';
13 import { ProductEditComponent } from './product-edit.component';
14 import { ProductEditGuard } from './product-edit.guard';
15
16 @NgModule({
17   imports: [
18     SharedModule,
19     ReactiveFormsModule,
20     InMemoryWebApiModule.forRoot(ProductData),
21     RouterModule.forChild([
22       { path: 'products', component: ProductListComponent },
23       { path: 'products/:id', component: ProductDetailComponent },
24       {
25         path: 'products/:id/edit',
26         canDeactivate: [ProductEditGuard],
27         component: ProductEditComponent
28       }
29     ])
30   ],
31   declarations: [
32     ProductListComponent,
33     ProductDetailComponent,
34     ProductEditComponent
35   ]
36 })

```

Table of Contents

- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Introduction
- Sample Application
- Sample Application: Architecture
- Sample Application: Code
- Routing to a Form
- Reading a Route Parameter
- Setting a canDeactivate Guard
- Refactoring to a Custom Validator
- Checklists and Summary
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Introduction
- Data Access Service
- Data Access Service: Demo

**Custom Validator:** We will create a custom validator in a separate file that can be used by other applications as well in the same project.

The screenshot shows a Linux desktop environment with a dark theme. A window titled "Angular Reactive Form" is open in Google Chrome, displaying a code snippet for a custom validator. The code defines a function named `range` that takes two parameters: `min` and `max`. It returns a validator function for an `AbstractControl` that checks if the value is between `min` and `max`. If the value is outside this range, it returns an object with a `range` key set to `true`. Otherwise, it returns `null`. The browser's sidebar shows a "Table of Contents" for a Pluralsight course on Angular Reactive Forms.

```
function range(min:number, max:number): ValidatorFn {
  return (c: AbstractControl): {[key: string]: boolean} | null => {
    if (c.value < min || c.value > max) {
      return { 'range': true };
    }
    return null;
  }
}
```

The method is made static so that instance of this class not required to be created

The screenshot shows a Visual Studio Code window with a dark theme. A file named `number.validator.ts` is open, containing a TypeScript code snippet. The `range` method is highlighted with a red box. The code imports `AbstractControl` and `ValidatorFn` from `@angular/forms`. It exports a class `NumberValidators` with a static `range` method that performs the same logic as the previous code but is marked as static. The Visual Studio Code interface includes a status bar at the bottom showing file details like "master" and "TypeScript 3.0.3". The sidebar on the right shows the same "Table of Contents" as the previous screenshot.

```
import { AbstractControl, ValidatorFn } from '@angular/forms';

export class NumberValidators {
  static range(min: number, max: number): ValidatorFn {
    return (c: AbstractControl): {[key: string]: boolean} | null => {
      if (c.value && isNaN(c.value) || c.value < min || c.value > max) {
        return { 'range': true };
      }
      return null;
    };
  }
}
```

## Create, Update, Read and Delete

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form +

May 7 21:43 ●

Angular Reactive Forms Create, Read, Update, and Delete (CRUD) Using HTTP : Data Access Service

Why Build a Data Access Service?

Separation of Concerns

Reusability

Data Sharing

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 15m 28s
- Template-driven vs. Reactive Forms 32m
- Building a Reactive Form 29m 36s
- Validation 35m 54s
- Reacting to Changes 19m 57s
- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s

Introduction 2m 10s

Data Access Service 3m 15s

Data Access Service: Demo 2m 7s

Faking a Backend Server 4m 57s

Populating the Form with Data 2m 57s

Populating the Form with Data 2m 12s

ADD NOTE

1:16 / 3:15

CC 1.0x

Speaker icon

Share icon

Back, Forward, Stop, Home, Refresh, Address Bar, Search icon, Add Note icon, Table of Contents icon, Notes icon, Course Overview icon, Introduction icon, Template-driven vs. Reactive Forms icon, Building a Reactive Form icon, Validation icon, Reacting to Changes icon, Dynamically Duplicate Input Elements icon, Reactive Form in Context icon, Create, Read, Update, and Delete (CRUD) Using HTTP icon, Play icon, Volume icon, CC icon, 1.0x icon, Share icon.

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form +

May 7 21:45 ●

Angular Reactive Forms Create, Read, Update, and Delete (CRUD) Using HTTP : Data Access Service

Steps to Building a Data Access Service

Import the Angular Http Service

Create the data access service

Import Observable and the observable operators

Inject the Angular Http Service

Write the code to issue each Http request

Table of Contents Notes

- Course Overview 1m 42s
- Introduction 15m 28s
- Template-driven vs. Reactive Forms 32m
- Building a Reactive Form 29m 36s
- Validation 35m 54s
- Reacting to Changes 19m 57s
- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s

Introduction 2m 10s

Data Access Service 3m 15s

Data Access Service: Demo 2m 7s

Faking a Backend Server 4m 57s

Populating the Form with Data 2m 57s

Populating the Form with Data 2m 12s

ADD NOTE

1:16 / 3:15

CC 1.0x

Speaker icon

Share icon

Back, Forward, Stop, Home, Refresh, Address Bar, Search icon, Add Note icon, Table of Contents icon, Notes icon, Course Overview icon, Introduction icon, Template-driven vs. Reactive Forms icon, Building a Reactive Form icon, Validation icon, Reacting to Changes icon, Dynamically Duplicate Input Elements icon, Reactive Form in Context icon, Create, Read, Update, and Delete (CRUD) Using HTTP icon, Play icon, Volume icon, CC icon, 1.0x icon, Share icon.

Steps for creating a service:

1) Import HttpClientModule

```

1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { HttpClientModule } from '@angular/common/http';
4 import { RouterModule } from '@angular/router';
5
6 import { AppComponent } from './app.component';
7 import { WelcomeComponent } from './home/welcome.component';
8 import { ProductModule } from './products/product.module';
9
10 @NgModule({
11   declarations: [
12     AppComponent,
13     WelcomeComponent
14   ],
15   imports: [
16     BrowserModule,
17     HttpClientModule, // Red box here
18     RouterModule.forRoot([
19       { path: 'welcome', component: WelcomeComponent },
20       { path: '', redirectTo: 'welcome', pathMatch: 'full' },
21       { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
22     ]),
23     ProductModule
24   ],
25   bootstrap: [AppComponent]
26 })
27 export class AppModule {}
```

Table of Contents

- Course Overview
- Introduction
- Template-driven vs. Reactive Forms
- Building a Reactive Form
- Validation
- Reacting to Changes
- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Introduction
- Data Access Service
- Data Access Service: Demo
- Faking a Backend Server
- Populating the Form with Data
- Populating the Form with Data: Demo

## 2) Create a serperate service

```

1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
3
4 import { Observable, of, throwError } from 'rxjs';
5 import { catchError, tap, map } from 'rxjs/operators';
6
7 import { Product } from './product';
8
9 @Injectable({ // Red box here
10   providedIn: 'root',
11 })
12 export class ProductService {
13   private productsUrl = 'api/products';
14
15   constructor(private http: HttpClient) { }
16
17   getProducts(): Observable<Product[]> { ... }
18
19   getProduct(id: number): Observable<Product> { ... }
20
21   createProduct(product: Product): Observable<Product> { ... }
22
23   deleteProduct(id: number): Observable<{}> { ... }
24 }
```

Table of Contents

- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
- Introduction
- Data Access Service
- Data Access Service: Demo
- Faking a Backend Server
- Populating the Form with Data
- Populating the Form with Data: Demo
- Saving Edits
- Saving Edits: Demo
- Creating New Items
- Creating New Items: Demo
- Deleting an Existing Item
- Deleting an Existing Item: Demo
- Checklists and Summary
- Final Words

For Getting the Data:

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ May 7 21:55 ●

HTTP Get Request

```
product.service.ts
```

```
...
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ProductService {
  private baseUrl = 'www.myWebService.com/api/products';

  constructor(private http: HttpClient) { }

  getProduct(id: number): Observable<Product> {
    const url = `${this.baseUrl}/${id}`;
    return this.http.get<Product>(url);
  }
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ May 7 21:55 ●

Calling the Data Access Service

```
product-edit.component.ts
```

```
...
constructor(private productService: ProductService) { }
...

getProduct(id: number): void {
  this.productService.getProduct(id)
    .subscribe(
      (product: Product) => this.displayProduct(product),
      (error: any) => this.errorMessage = <any>error
    );
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Example:

Create a method in the service class

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ May 7 22:00 ●

product.service.ts - APM - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

```
12 export class ProductService {  
13   private productsUrl = 'api/products';  
14  
15   constructor(private http: HttpClient) { }  
16  
17   getProducts(): Observable<Product[]> { ...  
18 }  
19  
20   getProduct(id: number): Observable<Product> {  
21     if (id === 0) {  
22       return of(this.initializeProduct());  
23     }  
24     const url = `${this.productsUrl}/${id}`;  
25     return this.http.get<Product>(url)  
26       .pipe(  
27         tap(data => console.log('getProduct: ' + JSON.stringify(data))),  
28         catchError(this.handleError)  
29       );  
30 }  
31  
32   createProduct(product: Product): Observable<Product> { ...  
33 }  
34  
35   deleteProduct(id: number): Observable<{}> { ...  
36 }  
37  
38   updateProduct(product: Product): Observable<Product> { ...  
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   errorMessage = `Backend returned code ${err.status}: ${err.body.error}`;  
80   console.error(err);  
81   return throwError(errorMessage);  
82 }  
83  
84  
85   private initializeProduct(): Product {  
86     // Return an initialized object  
87     return {  
88       id: 0,  
89       productName: null,  
90       productCode: null,  
91       tags: [''],  
92       releaseDate: null,  
93       price: null,  
94       description: null,  
95       starRating: null,  
96       imageUrl: null  
97     };  
98   }  
99 }
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data... 3m 12s
  - Populating the Form with Data... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Ln 67, Col 4 Spaces: 2 UTF-8 CRLF TypeScript 3.0.3

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ May 7 21:57 ●

product.service.ts - APM - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

```
12 export class ProductService {  
13   private productsUrl = 'api/products';  
14  
15   constructor(private http: HttpClient) { }  
16  
17   getProducts(): Observable<Product[]> { ...  
18 }  
19  
20   getProduct(id: number): Observable<Product> {  
21     if (id === 0) {  
22       return of(this.initializeProduct());  
23     }  
24     const url = `${this.productsUrl}/${id}`;  
25     return this.http.get<Product>(url)  
26       .pipe(  
27         tap(data => console.log('getProduct: ' + JSON.stringify(data))),  
28         catchError(this.handleError)  
29       );  
30 }  
31  
32   createProduct(product: Product): Observable<Product> { ...  
33 }  
34  
35   deleteProduct(id: number): Observable<{}> { ...  
36 }  
37  
38   updateProduct(product: Product): Observable<Product> { ...  
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   errorMessage = `Backend returned code ${err.status}: ${err.body.error}`;  
80   console.error(err);  
81   return throwError(errorMessage);  
82 }  
83  
84  
85   private initializeProduct(): Product {  
86     // Return an initialized object  
87     return {  
88       id: 0,  
89       productName: null,  
90       productCode: null,  
91       tags: [''],  
92       releaseDate: null,  
93       price: null,  
94       description: null,  
95       starRating: null,  
96       imageUrl: null  
97     };  
98   }  
99 }
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data... 3m 12s
  - Populating the Form with Data... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Ln 67, Col 4 Spaces: 2 UTF-8 CRLF TypeScript 3.0.3

## Consuming the service in the Component

```

    TS product.service.ts          TS product-edit.component.ts *
  99   addTag(): void {
100     this.tags.push(new FormControl());
101   }
102
103   deleteTag(index: number): void {
104     this.tags.removeAt(index);
105     this.tags.markAsDirty();
106   }
107
108   getProduct(id: number): void {
109     this.productService.getProduct(id)
110       .subscribe(
111         (product: Product) => this.displayProduct(product),
112         (error: any) => this.errorMessage = <any>error
113       );
114   }
115
116   displayProduct(product: Product): void {
117     if (this.productForm) {
118       this.productForm.reset();
119     }
120     this.product = product;
121
122     if (this.product.id === 0) {
123       this.pageTitle = 'Add Product';
124     } else {
125       this.pageTitle = `Edit Product: ${this.product.productName}`;
126     }
127
128     // Update the data on the form
129     this.productForm.patchValue({
130       productName: this.product.productName,
131       productCode: this.product.productCode,
132       starRating: this.product.starRating,
133       description: this.product.description
134     });
135     this.productForm.setControl('tags', this.fb.array(this.product.tags || []));
136   }
137
138   deleteProduct(): void {
139     if (this.product.id === 0) {
140       // Don't delete, it was never saved.
141       this.onSaveComplete();
142     }

```

May 7 22:05 ●

File Edit Selection View Go Debug Terminal Help

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s

```

    TS product.service.ts          TS product-edit.component.ts *
  115
  116   displayProduct(product: Product): void {
  117     if (this.productForm) {
  118       this.productForm.reset();
  119     }
  120     this.product = product;
  121
  122     if (this.product.id === 0) {
  123       this.pageTitle = 'Add Product';
  124     } else {
  125       this.pageTitle = `Edit Product: ${this.product.productName}`;
  126     }
  127
  128     // Update the data on the form
  129     this.productForm.patchValue({
  130       productName: this.product.productName,
  131       productCode: this.product.productCode,
  132       starRating: this.product.starRating,
  133       description: this.product.description
  134     });
  135     this.productForm.setControl('tags', this.fb.array(this.product.tags || []));
  136   }
  137
  138   deleteProduct(): void {
  139     if (this.product.id === 0) {
  140       // Don't delete, it was never saved.
  141       this.onSaveComplete();
  142     }

```

May 7 22:05 ●

File Edit Selection View Go Debug Terminal Help

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s

Updating the Data:

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ + May 7 23:00 ●

## HTTP Put Request

```
product.service.ts
```

```
...  
export class ProductService {  
  private baseUrl = 'www.myWebService.com/api/products';  
  
  constructor(private http: HttpClient) {}  
  
  updateProduct(product: Product): Observable<Product> {  
    const headers = new HttpHeaders({ 'Content-Type': 'application/json' });  
  
    const url = `${this.baseUrl}/${product.id}`;  
    return this.http.put<Product>(url, product, { headers: headers });  
  }  
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
    - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular Reactive Forms ▾ + May 7 23:01 ●

## Calling the Data Access Service

```
product-edit.component.ts
```

```
editProduct: void {  
  this.productService.updateProduct(p)  
    .subscribe(  
      () => this.onSaveComplete(),  
      (error: any) => this.errorMessage = <any>error  
    );  
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
    - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Edit Example:

product.service.ts

```
37  createProduct(product: Product): Observable<Product> { ... }
38
39
40
41  deleteProduct(id: number): Observable<{}> { ... }
42
43
44
45
46
47  updateProduct(product: Product): Observable<Product> {
48    const headers = new HttpHeaders({ 'Content-Type': 'application/json' });
49    const url = `${this.productsUrl}/${product.id}`;
50    return this.http.put<Product>(url, product, { headers });
51    .pipe(
52      tap(() => console.log('updateProduct: ' + product.id)),
53      // Return the product on an update
54      map((product) => product),
55      catchError(this.handleError)
56    );
57
58
59
60
61
62
63
64
65
66
67
68
69  private handleError(err) { ... }
70
71
72
73  private initializeProduct(): Product { ... }
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 }
```

Table of Contents

- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
  - Introduction
  - Data Access Service
  - Data Access Service: Demo
  - Faking a Backend Server
  - Populating the Form with Data
  - Populating the Form with Data: Demo
  - Populating the Form with Data: Demo
  - Saving Edits
  - Saving Edits: Demo
  - Creating New Items
  - Creating New Items: Demo
  - Deleting an Existing Item
  - Deleting an Existing Item: Demo
  - Checklists and Summary
- Final Words

product-edit.component.ts

```
162    (error: any) => this.errorMessage = <any>error
163  );
164 } else {
165   this.productService.updateProduct(p)
166   .subscribe(
167     () => this.onSaveComplete(),
168     (error: any) => this.errorMessage = <any>error
169   );
170 } else {
171   this.onSaveComplete();
172 }
173 } else {
174   this.errorMessage = 'Please correct the validation errors.';
175 }
176 }
177 }
178
179 onSaveComplete(): void {
180   // Reset the form to clear the flags
181   this.productForm.reset();
182   this.router.navigate(['/products']);
183 }
184 }
```

Table of Contents

- Dynamically Duplicate Input Elements
- Reactive Form in Context
- Create, Read, Update, and Delete (CRUD) Using HTTP
  - Introduction
  - Data Access Service
  - Data Access Service: Demo
  - Faking a Backend Server
  - Populating the Form with Data
  - Populating the Form with Data: Demo
  - Populating the Form with Data: Demo
  - Saving Edits
  - Saving Edits: Demo
  - Creating New Items
  - Creating New Items: Demo
  - Deleting an Existing Item
  - Deleting an Existing Item: Demo
  - Checklists and Summary
- Final Words

## Creating a new Field:

Activities Google Chrome ▾  
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ + May 7 23:18 ●

## Initializing an Object

```
product.service.ts
```

```
initializeProduct(): Product {
  return {
    id: 0,
    productName: null,
    productCode: null,
    tags: [''],
    releaseDate: null,
    price: null,
    description: null,
    starRating: null,
    imageUrl: null
  }
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾  
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ + May 7 23:20 ●

## HTTP Post Request

```
product.service.ts
```

```
...
export class ProductService {
  private baseUrl = 'www.myWebService.com/api/products';

  constructor(private http: HttpClient) { }

  createProduct(product: Product): Observable<Product> {
    const headers = new HttpHeaders({ 'Content-Type': 'application/json' });

    return this.http.post<Product>(this.baseUrl, product,
      { headers: headers });
  }
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Example:

```

152
153     saveProduct(): void {
154       if (this.productForm.valid) {
155         if (this.productForm.dirty) {
156           const p = { ...this.product, ...this.productForm.value };
157
158           if (p.id === 0) {
159             this.productService.createProduct(p)
160               .subscribe(
161                 () => this.onSaveComplete(),
162                 (error: any) => this.errorMessage = <any>error
163               );
164           } else {
165             this.productService.updateProduct(p)
166               .subscribe(
167                 () => this.onSaveComplete(),
168                 (error: any) => this.errorMessage = <any>error
169               );
170           }
171         } else {
172           this.onSaveComplete();
173         }
174       } else {
175         this.errorMessage = 'Please correct the validation errors.';
176       }
177     }

```

```

36
37   createProduct(product: Product): Observable<Product> {
38     const headers = new HttpHeaders({ 'Content-Type': 'application/json' });
39     product.id = null;
40     return this.http.post<Product>(this.productsUrl, product, { headers: headers })
41       .pipe(
42         tap(data => console.log('createProduct: ' + JSON.stringify(data))),
43         catchError(this.handleError)
44       );
45   }
46
47   deleteProduct(id: number): Observable<{}> { ...
48
49   updateProduct(product: Product): Observable<Products> {
50     const headers = new HttpHeaders({ 'Content-Type': 'application/json' });
51     const url = `${this.productsUrl}/${product.id}`;
52     return this.http.put<Product>(url, product, { headers: headers })
53       .pipe(
54         tap(() => console.log('updateProduct: ' + product.id)),
55         // Return the product on an update
56         map((product),
57           catchError(this.handleError)
58         );
59       );
60   }
61
62   private handleError(error: any): Observable<never> {
63     // ...
64   }

```

## Deleting:

Here if we notice in the below method `Observable<{}>` has a curly braces inside as delete does not return anything as the object is already deleted.

Activities Google Chrome ▾  
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ + May 7 23:26

## HTTP Delete Request

```
product.service.ts
```

```
...  
export class ProductService {  
  private baseUrl = 'www.myWebService.com/api/products';  
  
  constructor(private http: HttpClient) {}  
  
  deleteProduct(id: number): Observable<{}> {  
    const headers = new HttpHeaders({ 'Content-Type': 'application/json' });  
  
    const url = `${this.baseUrl}/${id}`;  
    return this.http.delete<Product>(url, { headers: headers });  
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾  
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ + May 7 23:28

## Calling the Data Access Service

```
product-edit.component.ts
```

```
deleteProduct: void {  
  this.productService.deleteProduct(this.product.id)  
    .subscribe(  
      () => this.onSaveComplete(),  
      (error: any) => this.errorMessage = <any>error  
    );  
}
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Example:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form +

May 7 23:29 ●

<https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m8&clip=12&mode=live>

Angular Reactive Forms

Create, Read, Update, and Delete (CRUD) Using HTTP : Deleting an Existing Item: Demo

product-edit.component.ts - APM - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

TS product.service.ts TS product-edit.component.ts < product-edit.component.html

```

137 deleteProduct(): void {
138   if (this.product.id === 0) {
139     // Don't delete, it was never saved.
140     this.onSaveComplete();
141   } else {
142     if (confirm(`Really delete the product: ${this.product.productName}?`)) {
143       this.productService.deleteProduct(this.product.id)
144         .subscribe(
145           () => this.onSaveComplete(),
146           (error: any) => this.errorMessage = <any>error
147         );
148     }
149   }
150 }
151
152 saveProduct(): void { ... }
153
154 onSaveComplete(): void {
155   // Reset the form to clear the flags
156   this.productForm.reset();
157   this.router.navigate(['/products']);
158 }
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185

```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF TypeScript 3.0.3 ⚡ ADD NOTE 🔍 CC 1.0x ⚙️ ↻

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 9s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data:... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s

## Checklist:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form +

May 7 23:31 ●

<https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m8&clip=13&mode=live>

Angular Reactive Forms

Create, Read, Update, and Delete (CRUD) Using HTTP : Checklists and Summary

## CRUD Checklist: Import the Http Service

app.module.ts

```

...
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  imports: [ HttpClientModule ],
  ...
})
export class AppModule { }

```

Add HttpClientModule to the imports array of an Angular Module

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF TypeScript 3.0.3 ⚡ ADD NOTE 🔍 CC 1.0x ⚙️ ↻

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data:... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾ May 7 23:31 ●

Google News Home | Pluralsight Angular Reactive Form +

Angular Reactive Forms Create, Read, Update, and Delete (CRUD) Using HTTP : Checklists and Summary

## CRUD Checklist: Data Access Service

**Import what we need**

**Define a dependency for the http client service**

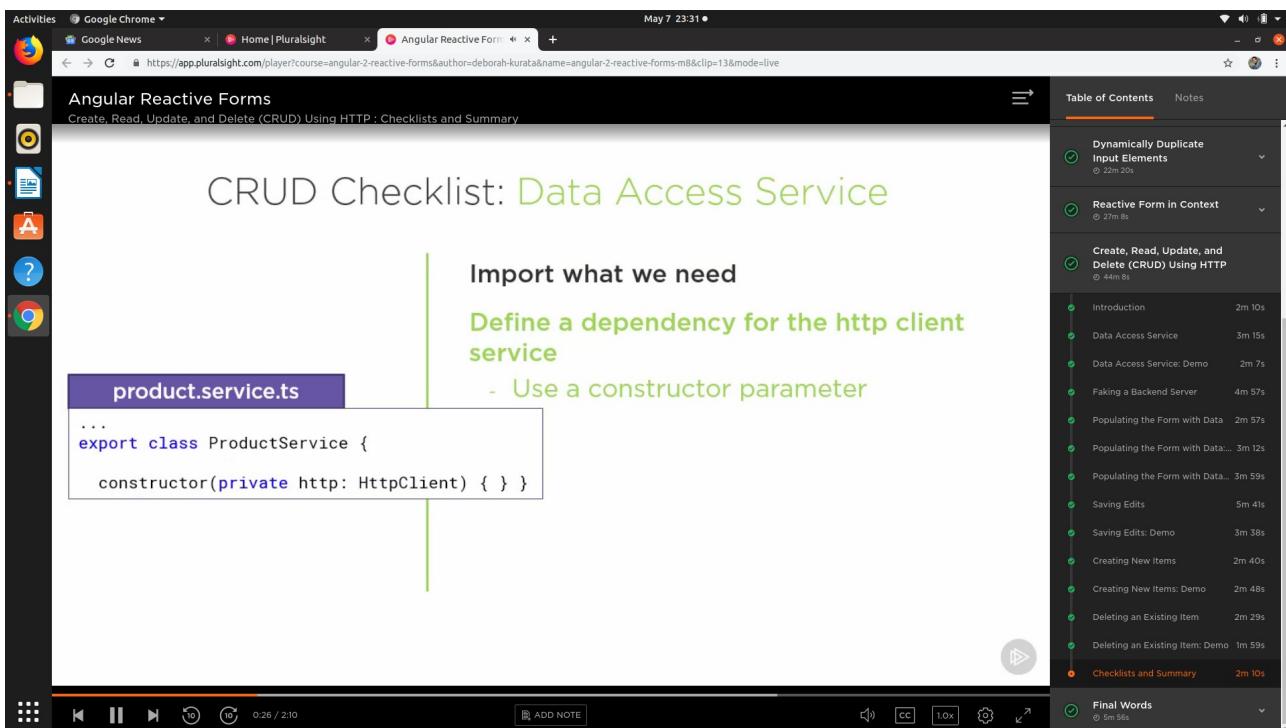
- Use a constructor parameter

```
product.service.ts
...
export class ProductService {
  constructor(private http: HttpClient) { } }
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data:... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 58s

◀ ▶ ⏪ ⏩ 0:26 / 2:10 ADD NOTE 🔍 CC 1.0x ⚙️ ⌂



Activities Google Chrome ▾ May 7 23:32 ●

Google News Home | Pluralsight Angular Reactive Form +

Angular Reactive Forms Create, Read, Update, and Delete (CRUD) Using HTTP : Checklists and Summary

## CRUD Checklist: Data Access Service

**Import what we need**

**Define a dependency for the http client service**

- Use a constructor parameter

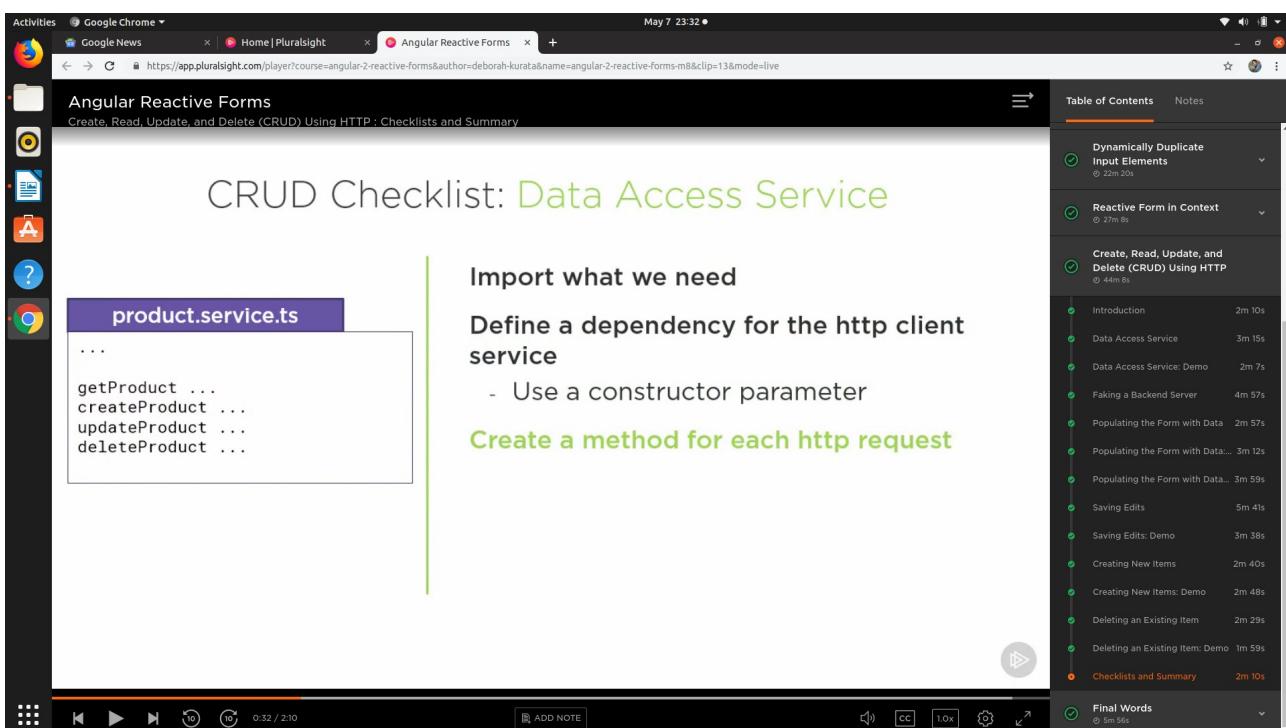
**Create a method for each http request**

```
product.service.ts
...
getProduct ...
createProduct ...
updateProduct ...
deleteProduct ...
```

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data:... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 58s

◀ ▶ ⏪ ⏩ 0:32 / 2:10 ADD NOTE 🔍 CC 1.0x ⚙️ ⌂



Activities Google Chrome ▾ May 7 23:32 ●

Google News Home | Pluralsight Angular Reactive Form +

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m8&clip=13&mode=live

## CRUD Checklist: Data Access Service

**product.service.ts**

```
product.service.ts
...
const url =
  `${this.baseUrl}/${id}`;
return this.http.get(url);
```

**Import what we need**

**Define a dependency for the http client service**

- Use a constructor parameter

**Create a method for each http request**

**Call the desired http method, such as get**

- Pass in the Url

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data:... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾ May 7 23:32 ●

Google News Home | Pluralsight Angular Reactive Form +

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m8&clip=13&mode=live

## CRUD Checklist: Data Access Service

**product.service.ts**

```
product.service.ts
...
const url =
  `${this.baseUrl}/${id}`;
return this.http.get<Product>(url)
  .pipe(
    catchError(this.handleError)
  );
```

**Import what we need**

**Define a dependency for the http client service**

- Use a constructor parameter

**Create a method for each http request**

**Call the desired http method, such as get**

- Pass in the Url

**Add error handling**

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
- Introduction 2m 10s
- Data Access Service 3m 15s
- Data Access Service: Demo 2m 7s
- Faking a Backend Server 4m 57s
- Populating the Form with Data 2m 57s
- Populating the Form with Data:... 3m 12s
- Populating the Form with Data:... 3m 59s
- Saving Edits 5m 41s
- Saving Edits: Demo 3m 38s
- Creating New Items 2m 40s
- Creating New Items: Demo 2m 48s
- Deleting an Existing Item 2m 29s
- Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Form +

May 7 23:32 •

## CRUD Checklist: Using the Service

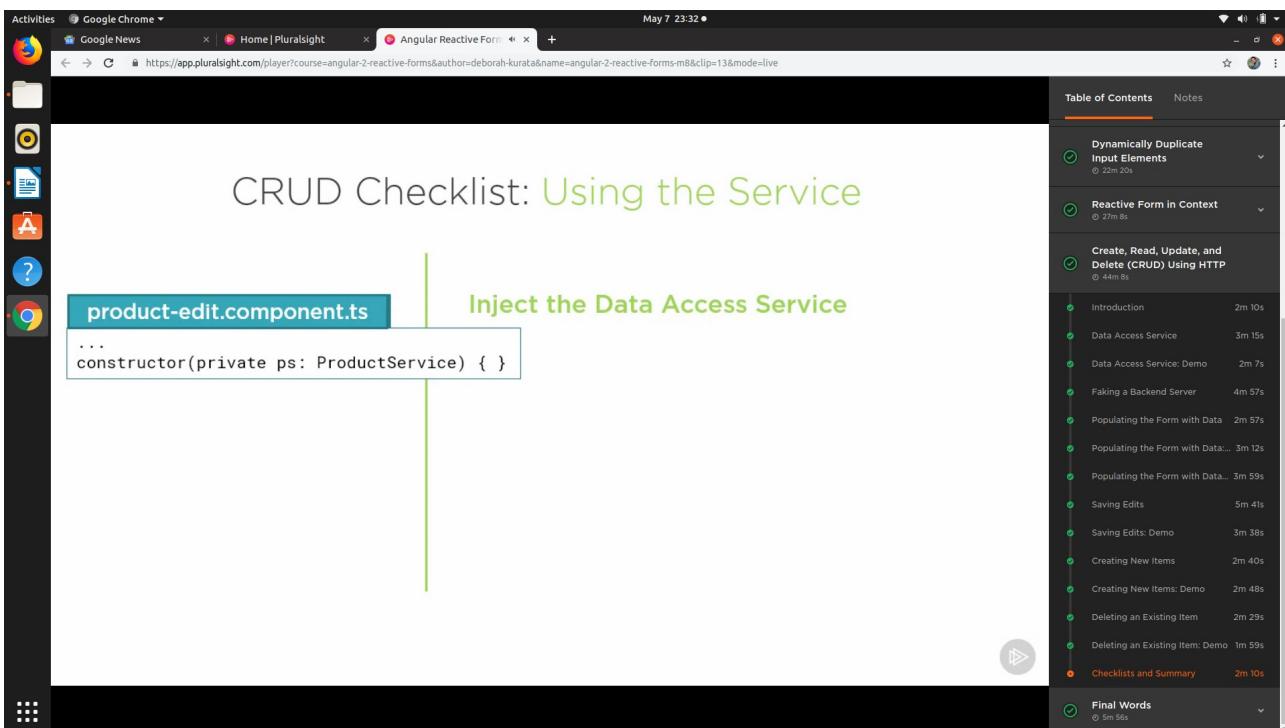
product-edit.component.ts

```
...  
constructor(private ps: ProductService) { }
```

Inject the Data Access Service

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s



Activities Google Chrome ▾

Google News Home | Pluralsight Angular Reactive Forms +

May 7 23:33 •

## CRUD Checklist: Using the Service

product-edit.component.ts

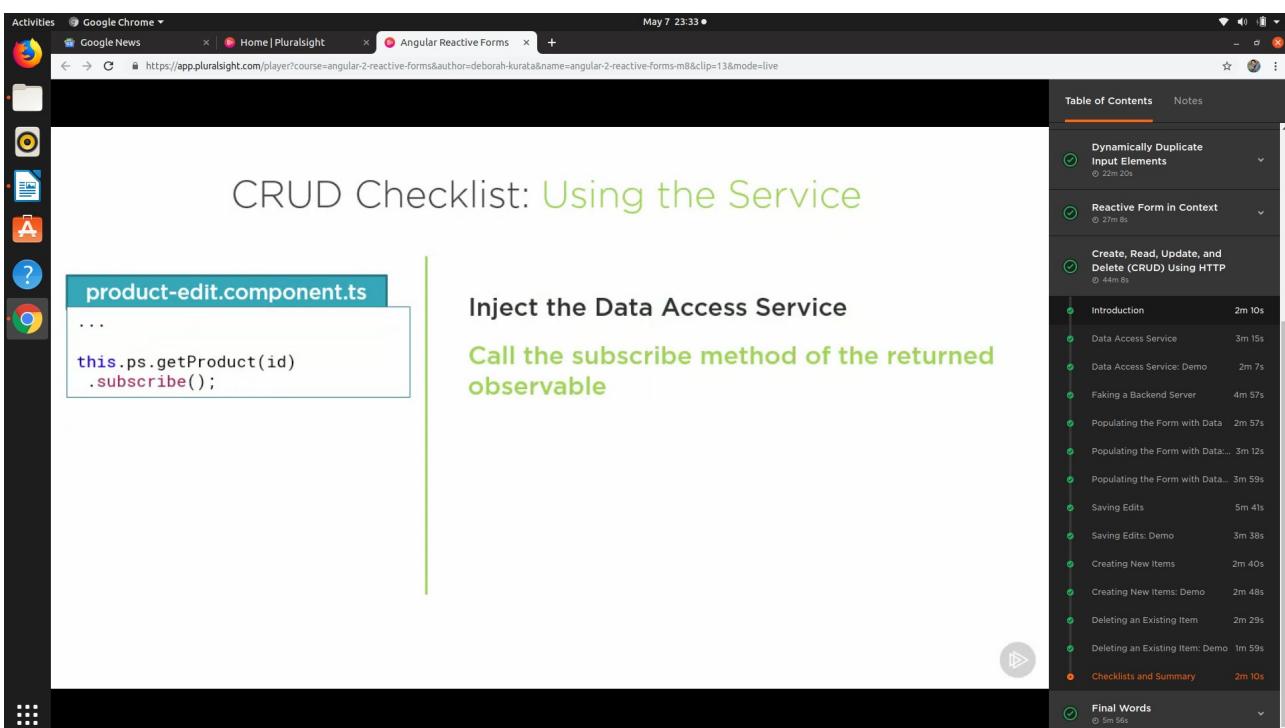
```
...  
  
this.ps.getProduct(id)  
.subscribe();
```

Inject the Data Access Service

Call the subscribe method of the returned observable

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
- Checklists and Summary 2m 10s
- Final Words 5m 56s



Activities Google Chrome ▾ May 7 23:33 ●

Google News Home | Pluralsight Angular Reactive Form +

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m8&clip=13&mode=live

## CRUD Checklist: Using the Service

```
product-edit.component.ts
...
this.ps.getProduct(id)
.subscribe(
  (product: Product) =>
    this.onRetrieved(product)
);
```

**Inject the Data Access Service**

**Call the subscribe method of the returned observable**

**Provide a function to handle an emitted item**

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾ May 7 23:33 ●

Google News Home | Pluralsight Angular Reactive Form +

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m8&clip=13&mode=live

## CRUD Checklist: Using the Service

```
product-edit.component.ts
...
this.ps.getProduct(id)
.subscribe(
  (product: Product) =>
    this.onRetrieved(product),
  (error: any) =>
    this.errorMessage = error
);
```

**Inject the Data Access Service**

**Call the subscribe method of the returned observable**

**Provide a function to handle an emitted item**

**Provide an error function to handle any returned errors**

Table of Contents Notes

- Dynamically Duplicate Input Elements 22m 20s
- Reactive Form in Context 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 44m 8s
  - Introduction 2m 10s
  - Data Access Service 3m 15s
  - Data Access Service: Demo 2m 7s
  - Faking a Backend Server 4m 57s
  - Populating the Form with Data 2m 57s
  - Populating the Form with Data:... 3m 12s
  - Populating the Form with Data:... 3m 59s
  - Saving Edits 5m 41s
  - Saving Edits: Demo 3m 38s
  - Creating New Items 2m 40s
  - Creating New Items: Demo 2m 48s
  - Deleting an Existing Item 2m 29s
  - Deleting an Existing Item: Demo 1m 59s
  - Checklists and Summary 2m 10s
- Final Words 5m 56s

Activities Google Chrome ▾  
Google News ▾ Home | Pluralsight ▾ Angular Reactive Form ▾ + May 7 23:40 ●

https://app.pluralsight.com/player?course=angular-2-reactive-forms&author=deborah-kurata&name=angular-2-reactive-forms-m9&clip=2&mode=live



## Learning More

**Intermediate Angular Courses**

- Angular Routing
- Angular HTTP Communication
- Angular Component Communication

Table of Contents Notes

- Course Overview 0m 42s
- Introduction 0m 28s
- Template-driven vs. Reactive Forms 0m 52m
- Building a Reactive Form 0m 39m 36s
- Validation 0m 35m 54s
- Reacting to Changes 0m 37m 57s
- Dynamically Duplicate Input Elements 0m 22m 20s
- Reactive Form in Context 0m 27m 8s
- Create, Read, Update, and Delete (CRUD) Using HTTP 0m 44m 55s
- Final Words 0m 56s
- Introduction 0m 28s
- Recapping Your Journey 4m 6s
- Learning More 0m 59s
- Closing 0m 22s