Model-driven Forms

By: Mosh Hamedani

New Module

In the last section, you learned about template-driven forms. With such forms, we mostly focus on our HTML templates and let Angular create Control objects under the hood.

Template-driven forms are easy to create but they give us limited control over validation. To implement custom validation, we need to create controls explicitly. And this section is all about this.

In order to create controls explicitly, you need to import **ReactiveFormsModule** (in addition to **FormsModule**) in **app.module.ts:**

```
import { FormsModule, ReactiveFormsModule } from '@angular/forms';


@NgModule({
    imports: [
        FormsModule,
        ReactiveFormsModule
        …
    ]
})
```

Breaking Changes

Some of the form-related classes are renamed in the final release. Here is a cheat sheet of these changes:

Import statements

Beta	`import { Control, ControlGroup } from 'angular2/common';`
Final	`import { FormControl, FormGroup } from '@angular/forms';`

Model-driven Forms

By: Mosh Hamedani

Declaring forms

Beta	`<form [ngFormModel]="form">`
Final	`<form [formGroup]="form">`

Associating input fields with explicitly-created control objects

Beta	`<input ngControl="username">`
Final	`<form formControlName="username">`

Beta	`<div ngControlGroup="billing">`
Final	`<form formGroupName="billing">`

Accessing the underlying control object

Beta	`<input #username="ngForm">`
	…
	`<div *ngIf="!username.valid>Error</div>`

Final	`<form [formGroup]="form">`
	…
	`<div *ngIf="!form.controls.username.valid">Error</div>`

Accessing a control inside a component (in lecture Validating Upon Submit):

Beta	`this.form.find('username')`
Final	`this.form.controls['username']`