Forms Breaking Changes

By: Mosh Hamedani

Template-driven forms

We use template-driven forms for building data entry forms with simple validation. In final release, we have a module specifically designed for working with forms. In order to build forms with Angular, you must import this module into **app.module.ts** (or any modules where you have a form)**:**

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


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

In Angular Beta, we applied **ngControl** directive on our input fields. Angular would create control objects and link them to the corresponding input fields under the hood. This control object represents the state of an input field (e.g. whether it is valid or not, whether its value is changed or not, etc). Here's an example:

```
<input ngControl="firstName">
```

This syntax is changed in final release. Instead of using **ngControl**, we use **ngModel.** We should also set the **name** attribute.

```
<input ngModel name="firstName">
```

Beta	`<input ngControl="firstName">`
Final	`<input ngModel name="firstName">`

Forms Breaking Changes

By: Mosh Hamedani

To display validation errors, we need to access the underlying control object. In Angular Beta, we declared a template variable and set it to **ngForm**:

```
<input ngControl="firstName" #firstName="ngForm">
```

In final release, we set our template variable to **ngModel** instead of **ngForm.**

Beta	`<input … #firstName="ngForm">`
Final	`<input … #firstName="ngModel">`

Model-driven forms

Template-driven forms are easy to create but they give us limited control over validation. To implement custom validation, we need to use model-driven forms. And this means, we need to create control objects explicitly.

In final release, we should import **ReactiveFormsModule** (in addition to **FormsModule**) in **app.module.ts** (or any modules that includes a model-driven form)**:**

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


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

Forms Breaking Changes

By: Mosh Hamedani

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';`

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):

Forms Breaking Changes

By: Mosh Hamedani

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