New Forms Module

By: Mosh Hamedani

There are a few simple changes (mostly syntactical or name changes) in Angular forms. All form classes have been moved from '**@angular/common**' to a new library: '**@angular/forms**'.

In order to use the new form APIs we need to import the new **FormsModule** into our application module (or any modules that require form functionality).

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


@NgModule({
    imports: [
        FormsModule
    ]
})
export class AppModule {}
```

Template-driven Forms

Here is a cheat sheet of changes. Continue reading for more details:

Old API	New API
`<input ngControl="firstName">`	`<input name="firstName" ngModel>`
`<div ngControlGroup="address">`	`<div ngModelGroup="address">`
`<input #firstName="ngForm">`	`<input #firstName>`

ngModel

Previously, we used **ngControl** directive on input fields. Angular would create a form control object for these input fields under the hood:

```
<input type="text" ngControl="firstName" />
```

Now, instead of **ngControl**, we apply **ngModel** directive along with the **name** attribute.

```
<input type="text" name="firstName" ngModel />
```

Note that we're simply applying **ngModel** directive on the input field, and we don't necessarily have to use the two-way binding syntax (unless we want to bind this input field to a component property).

What happens if we forget to apply **ngModel** directive on an input field? Then, we won't have a form control object for that field. So, if we log the JSON object that represents the form, we won't see that field.

ngModelGroup

If a few fields are part of a group, we use **ngModelGroup** directive:

```
<fieldset ngModelGroup="address">
    <input type="text" name="city" ngModel>
    <input type="text" name="zip" ngModel>
</fieldset>
```

With this, in the submit handler, we'll get a nested structure like this:

```
{
    address: {
        city: “Melbourne”,
        zip: 3144
    }
}
```

Referencing form controls

To get a reference to the underlying form control object, previously we used the following syntax:

```
<input #name=“ngForm” />
```

We could use this reference to display validation errors:

```
<div *ngIf=“name.errors”>…</div>
```

This has been simplified in the API. We simply assign a template variable to an input field, without setting it to “ngForm”, and this will reference the underlying form control object:

```
<input #name />
```

New Forms Module

By: Mosh Hamedani

Model-driven Forms

We need to import **ReactiveFormsModule** (in addition to **FormsModule**) into our module:

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


@NgModule({
    imports: [
        FormsModule,
        ReactiveFormsModule
    ]
})
export class AppModule {}
```

Note that this is just an experimentation and it's likely that in the final release, ReactiveFormsModule will be merged with FormsModule.

There are a few name changes in the directives:

Old API	New API
`<form [ngFormModel]="registerForm">`	`<form [formGroup]="registerForm">`
`<input ngControl="firstName">`	`<input formControlName="firstName">`
`<fieldset ngControlGroup="address">`	`<fieldset formGroupName="address">`

Everything else is exactly the same.