Angular Forms

By: Mosh Hamedani

Model-driven forms

In the component:

```
import
   {ControlGroup, FormBuilder, Validators}
   from ‘angular2/common’;


signupForm: ControlGroup;


constructor(fb: FormBuilder){
     this.signupForm = fb.group({
         name: [‘’, Validators.required],
         email: [],
         billing: fb.group({
             cardNumber: [‘’, Validators.required],
             expiry: [‘’, Validators.required]
         })
     });
}
```

In the template:

```
<form [ngFormModel]="signupForm">
    …
    <input ngControl="name">
    …
    <input ngControl="email">
    …
    <div ngControlGroup="billing">
        <input ngControl="cardNumber">
        …
        <input ngControl="expiry">
    </div>
</form>
```

Implementing Custom Validators

```
export class UsernameValidators {
    static cannotContainSpace(control: Control) {
         … // validation logic;

         if (invalid)
            return { cannotContainSpace: true });

         return null;
    }
}
```

When creating the control using FormBuilder:

```
name: [‘’, Validators.compose([
    Validators.required,
    UsernameValidators.cannotContainSpace
])]
```

Async Validator

The same as a custom validator, but we return a Promise.

```
static shouldBeUnique(control: Control){
     return new Promise((resolve, reject) => {
         … // validation logic
         if (invalid)
             resolve({ shouldBeUnique: true });
         else
             resolve(null);
     });
}
```

When building a control using FormBuilder:

```
name: [‘’,Validators.required, UsernameValidators.shouldBeUnique]
```

Note the syntax:

```
[defaultValue, customValidators, asyncValidators]
```

When using more than one custom or async validator, we use the **compose** or **composeAsync** methods:

```
[
    defaultValue,
    Validators.compose([v1, v2]),
    Validators.composeAsync([v3, v4])
];
```

To show a loader when an async validator is in progress:

```
<input ngControl="name">
<div *ngIf="name.control.pending">Checking for uniqueness…</div>
```