Formularios en Angularjs

Anxo Fole
afole@plainconcepts.com

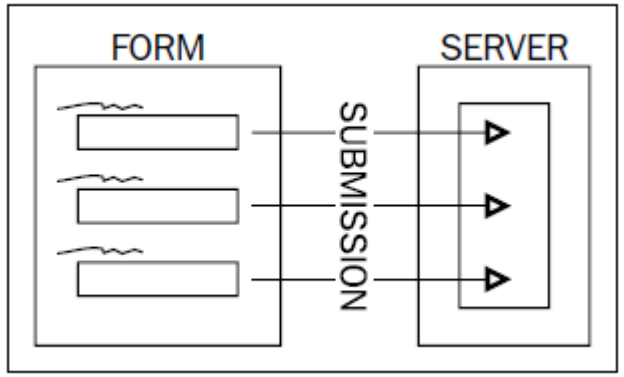
Función

- Recogen diferentes entradas de datos del usuario para enviarlas al servidor
- Tradicionalmente no ha existido un mecanismo de validación integrado
- Con Angular, controles (input, select, textarea) y formularios proporcionan servicios de validación y podemos notificar al usuario antes de que envie el formulario.
  - Mejor experiencia de usuario que la validación en servidor: Feedback
  - La validación en servidor aún es necesaria.

Modelo tradicional vs dataBinding

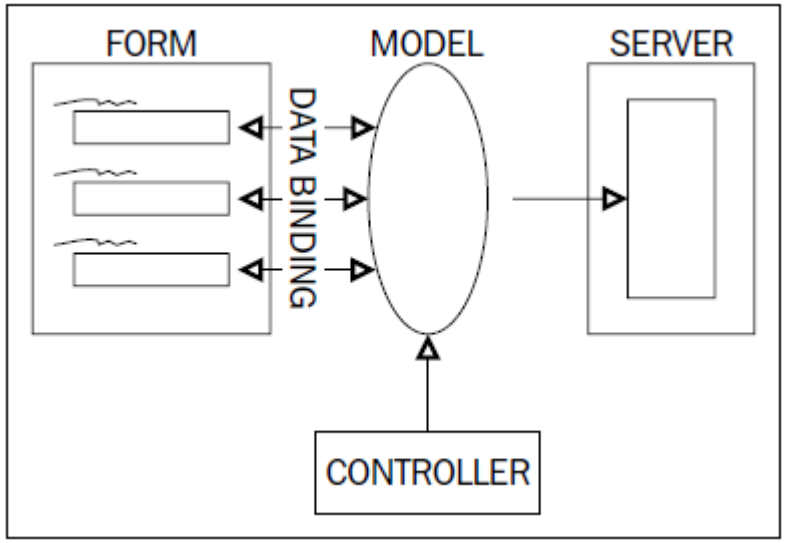
Tradicional

- Envío de datos en ***submit*** del formulario a la url definida en el atributo ***action***



DataBinding

- Después del binding con el modelo utiliza XHR para el envío



Formulario

```
<form novalidate>
<label>E-mail</label>
<input type="email"
       ng-model="user.email"
       name="email" required>
<label>
       <input type="radio"
       ng-model="user.preference"
       value="emacs" />emacs</label><br />


<input type="button" ng-click="reset()" value="Reset" />
<input type="submit" ng-click="update(user)" value="Save" />

</form>
```

Directivas

- **form:** ngFormController
  - **ngForm:** formulario anidado. Siempre dentro de un form. No puede sustituirlo.
- **ngModel:** binding bidireccional
  - **ngModelOptions:** configura opciones de actualización del modelo
- **input:** ngModelController
- **textarea:** ngModelController
- **select:** Combos bindeados arrays de objetos
  - **ngSelected:** marca como seleccionado

- **Otras:** ngChange, ngReadonly, ngValue, ngChecked, …

ngModelOptions

- **updateOn:** evento que desencadena la actualización del modelo
  - Pueden ser más de uno
- **debounce:** delay en milisegundos que tardará en actualizarse el modelo.
  - Se puede especificar un objeto con valores distintos para diferentes eventos
- **allowInvalid:** el modelo se puede actualizar con valores no válidos

```
ng-model-options="{ updateOn: 'default blur', debounce: {'default': 500, 'blur': 0}, allowInvalid: true }"
```

Atributos

- Definir un **name** al formulario y a los inputs de un formulario
  - De esta forma están disponibles en el $scope
  - Pueden ser utilizados por otras directivas

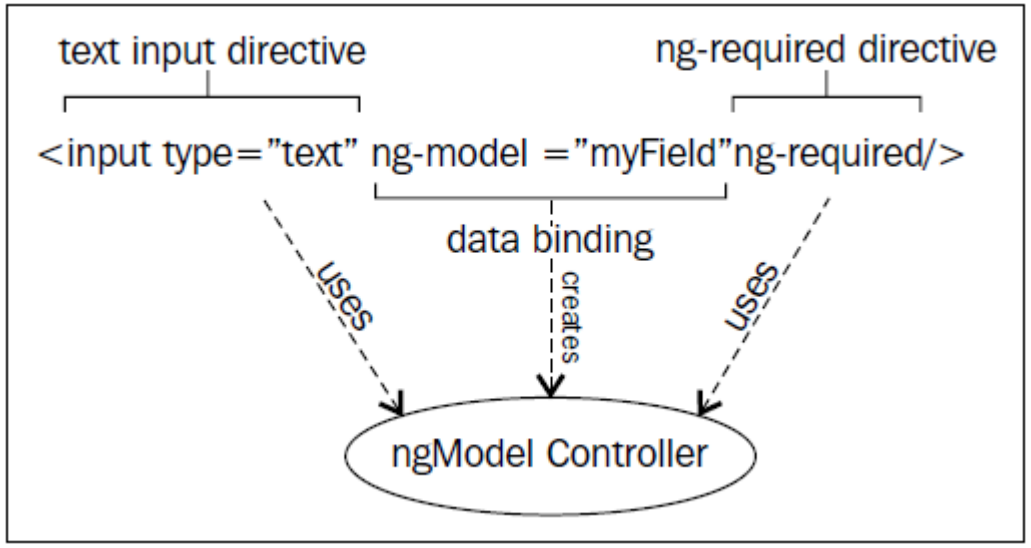
- Añadir el atributo **novalidate** al formulario
  - HTML5
  - Deshabilita la validación incluida en el navegador
  - Angular asume el control de las validaciones

Atributo name

HTML	Scope	Controller
	model1, model2, …	
<form name="form1">	form1 : {	ngFormController
	$valid, $invalid,	
	$pristine, $dirty, …	
<input	field1: {	ngModelController
name="field1"	$valid, $invalid,	
ng-model="model1"	$pristine, $dirty, …	
/>	},	
<input	field2: {	ngModelController
name="field2"	$valid, $invalid,	
ng-model="model2"	$pristine, $dirty, …	
/>	}	
</form>	},	

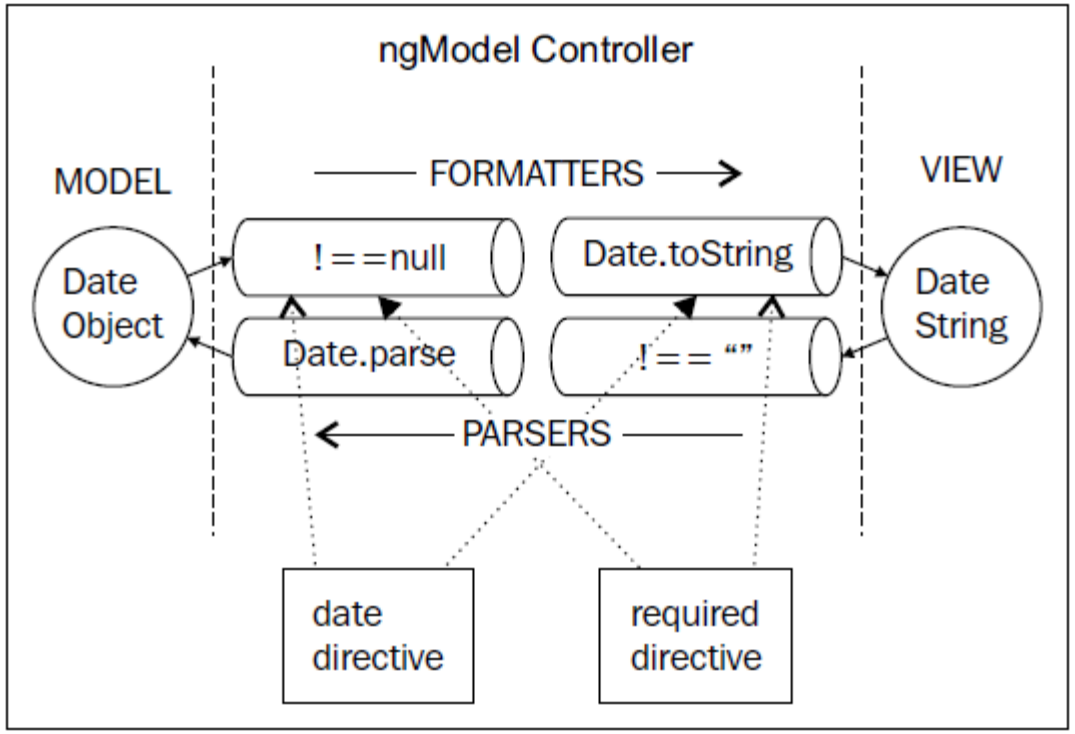
ngModelController

- Cada ng-model (+ input) crea una instancia de este controlador
  - La directiva define un controlador propio
- Es el responsable de gestionar el dataBinding bidireccional entre el modelo y el DOM
- También gestiona la validez del modelo (con otras directivas)



ngModelController

- Internamente define un pipeline de transformación
- *Del modelo a la vista*
  - **Formatters**: del modelo a la vista
- *De la vista al modelo*
  - **Parsers**: de la vista al modelo
  - **Validators:** tras los parsers, ejecutan funciones de validación
- Responsables de la lógica de validación



ngModelController: propiedades

- $modelValue
- $viewValue

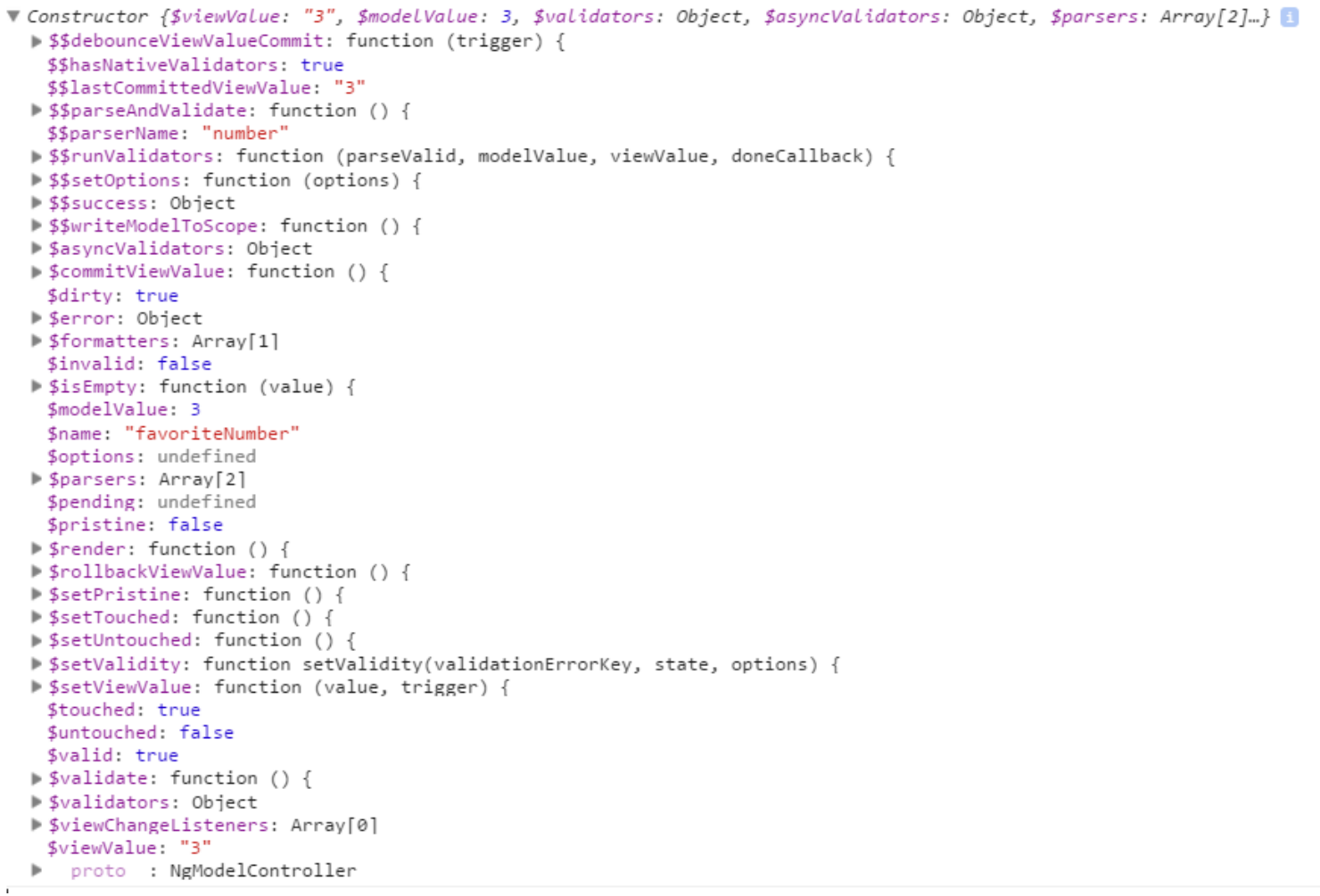
- $pristine
- $dirty
- $valid
- $invalid
- $touched
- $untouched

- $formatters
- $parsers
- $validators
- $asyncValidators
- $pending

- $rollbackViewValue

- $error (bool por key)

ngModelController



Actualización del modelo

- ***$viewValue*** a través de los parsers y validators asigna el ***$modelValue***
- El modelo no se actualiza mientras no se cumplan todas la reglas de validación
  - Mantiene consistente el modelo
  - Ahora se pueden deshabilitar los validadores (no los parsers) con ngModelOptions (allowInvalid: true)

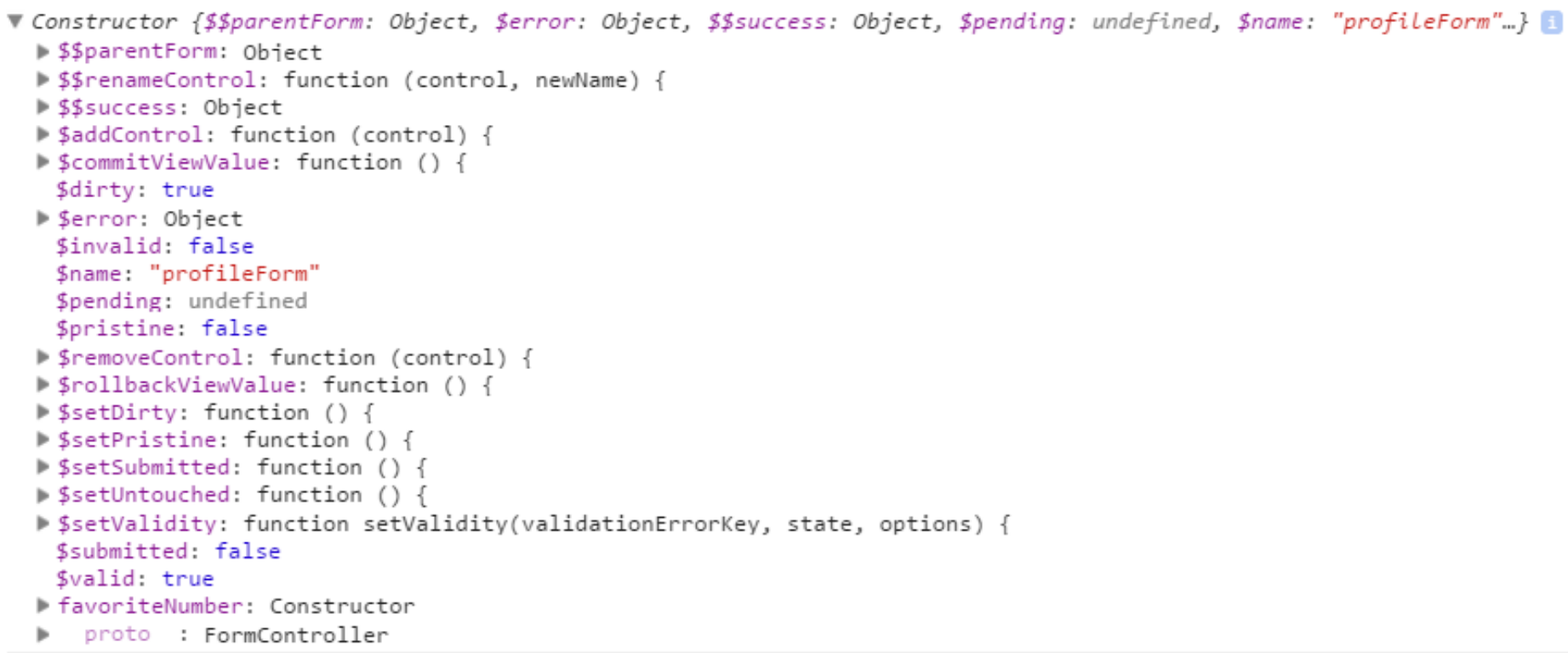
ngFormController

- Todos los inputs que se encuentran dentro de un form se registran en el formsController
- Ofrece acceso al estado agregado de validación y edición de todos los controles de edición a los que engloba

ngFormController: propiedades

- $pristine
- $dirty
- $valid
- $invalid
- $submitted
- $pending
- $rollbackViewValue
- $error (array por key)

ngFormController



Estilado de vista según estado

- Además de las propiedades de FormController y ModelController, se definen ***clases*** dinámicamente en ***form*** e ***inputs***.

Form

- ng-pristine
- ng-dirty
- ng-valid
- ng-invalid
- ng-submitted

Input

- ng-pristine
- ng-dirty
- ng-valid
- ng-invalid
- ng-touched
- ng-untouched

Propiedades y clases específicas por validador

- Cada validador añade su propiedad al objeto $error
  - Podemos saber qué regla concreta se está incumpliendo
- Genera también una clase específica para cada validador
  - Podemos personalizar cómo se muestra un elemento en función del error

- required
  - $error.required (en ModelController como bool y en FormController como array)
  - ng-valid-required o ng-invalid-required

$error

- email
- max
- maxlength
- min
- minlength
- number
- pattern
- required
- url
- date
- datetimelocal
- time
- week
- month

Directivas de validación

- Se soportan las validaciones básicas de HTML5
  - Input types: text, number, url, email, checkbox
- Además existen directivas de validación
  - ng-required
  - ng-pattern
  - ng-minLength
  - ng-maxLength
  - ng-min
  - ng-max

Mensajes de validación

- Utilizar ng-class, ng-show y ng-hide para proporcionar feedback al usuario sobre el estado de la validación

```
<form name="userInfoForm“ novalidate>
    <div class="control-group"
         ng-class="getCssClasses(userInfoForm.email)">
        <label>E-mail</label>
        <input type="email" ng-model="user.email"
               name="email" required>
        <span ng-show="showError(userInfoForm.email, 'email')" ...>
            You must enter a valid email
        </span>
        <span ng-show="showError(userInfoForm.email, 'required')" ...>
            This field is required
        </span>
    </div>
</form>
```

Mensajes de validación

```
app.controller('MainCtrl', function ($scope) {
    $scope.getCssClasses = function (ngModelContoller) {
        return {
            error: ngModelContoller.$invalid && ngModelContoller.$dirty,
            success: ngModelContoller.$valid && ngModelContoller.$dirty
        };
    };
    $scope.showError = function (ngModelController, error) {
        return ngModelController.$error[error];
    };
});
```

Mensajes de validación

- ***ng-messages*** y ***ng-message*** (módulo ***ngMessages***)

```
<form name="myForm">
    <label>Enter your name:</label>
    <input type="text"
           name="myName"
           ng-model="name"
           ng-minlength="5"
           ng-maxlength="20"
           required />

    <div ng-messages="myForm.myName.$error">
        <div ng-message="required">You did not enter a field</div>
        <div ng-message="minlength">Your field is too short</div>
        <div ng-message="maxlength">Your field is too long</div>
    </div>
</form>
```

Deshabilitar el botón de submit

```
<form name="userInfoForm">
    ...
    <button ng-disabled="!canSave()">Save</button>
</form>


app.controller('MainCtrl', function ($scope) {
    $scope.canSave = function () {
        return $scope.userInfoForm.$dirty &&
        $scope.userInfoForm.$valid;
    };
});
```

Envío del formulario

- Usar uno de los dos:
  - ng-submit como atributo del elemento **form** + un solo input type="submit" en el formulario (sin directivas específicas)
  - ng-click en el primer botón del formulario

Subformularios como elementos reusables

```
<script type="text/ng-template" id="password-form">
    <ng-form name="passwordForm">
        <div ng-show="user.password != user.password2">
            Passwords do not match
        </div>
        <label>Password</label>
        <input ng-model="user.password" type="password" required>
        <label>Confirm Password</label>
        <input ng-model="user.password2" type="password" required>
    </ng-form>
</script>
<form name="form1" novalidate>
    <legend>User Form</legend>
    <label>Name</label>
    <input ng-model="user.name" required>
    <ng-include src="'password-form'"></ng-include>
</form>
```