



# AngularJS

*Jour 9 - Formulaires & directives avancés*

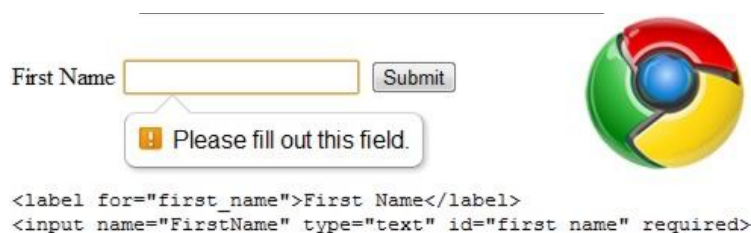
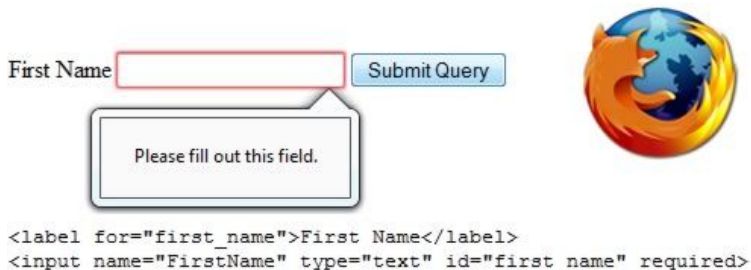
# Au programme !

- Les formulaires
- Les directives avancées
- Le testing

# 1 - Les formulaires

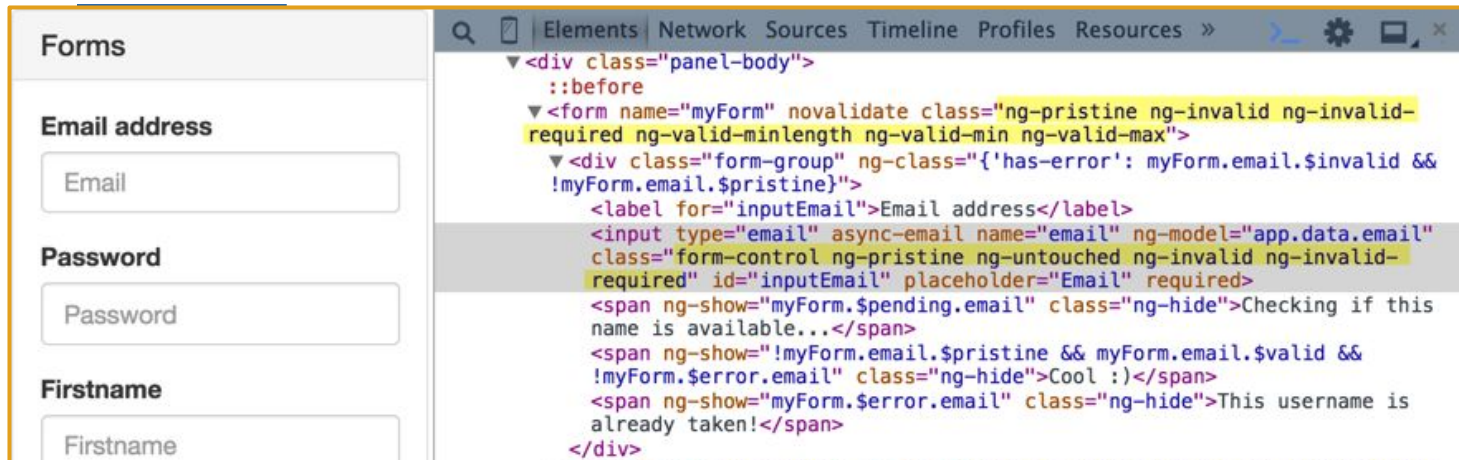
# Formulaire - Création d'un formulaire

- Avec `<form/>` ou `ng-form`
  - `<form name="myForm">`
  - `<div ng-form="myForm">`
- myForm** : Nom de la variable sur le scope
- Désactiver les validations natives
  - `<form name="myForm" novalidate></form>`



# Formulaire - Les classes CSS

- AngularJS rajoute des classes pour indiquer l'état du formulaire/champs
  - <https://docs.angularjs.org/guide/forms#using-css-classes>



The image shows a browser window with a form on the left and its HTML code in the developer tools on the right. The form has three input fields: 'Email address', 'Password', and 'Firstname'. The developer tools show the HTML structure of the form, with the 'Email address' input field highlighted. The code includes various AngularJS classes like 'ng-pristine', 'ng-invalid', and 'ng-invalid-required' to indicate the state of the form and its fields. The 'Email address' input field is currently in a 'pristine' state, as indicated by the 'ng-pristine' class.

```
<div class="panel-body">
  ::before
  <form name="myForm" novalidate class="ng-pristine ng-invalid ng-invalid-required ng-valid-minlength ng-valid-min ng-valid-max">
    <div class="form-group" ng-class="{ 'has-error': myForm.email.$invalid && !myForm.email.$pristine }">
      <label for="inputEmail">Email address</label>
      <input type="email" async-email name="email" ng-model="app.data.email" class="form-control ng-pristine ng-untouched ng-invalid ng-invalid-required" id="inputEmail" placeholder="Email" required>
      <span ng-show="myForm.$pending.email" class="ng-hide">Checking if this name is available...</span>
      <span ng-show="!myForm.email.$pristine && myForm.email.$valid && !myForm.$error.email" class="ng-hide">Cool :)</span>
      <span ng-show="myForm.$error.email" class="ng-hide">This username is already taken!</span>
    </div>
  </form>
</div>
```

# Formulaire - L'objet Formulaire

- Le formulaire publie un objet sur le scope
  - [http://localhost:8080/day\\_09/step\\_01/](http://localhost:8080/day_09/step_01/)
  - FormController
    - <https://docs.angularjs.org/api/ng/type/form.FormController>
  - ngModelController
    - <https://docs.angularjs.org/api/ng/type/ngModel.NgModelController>

## 2 - Les Directives avancées

# Directive - L'option require

- require: "ngModel"  
→ Ma directive nécessite *ng-model*

```
<input type="text" ma-directive />
```

```
<input type="text" ng-model="data.user" ma-directive />
```

- require: "^ngModel"  
→ Ma directive nécessite ng-model en parent

```
<div><input type="text" ma-directive /></div>
```

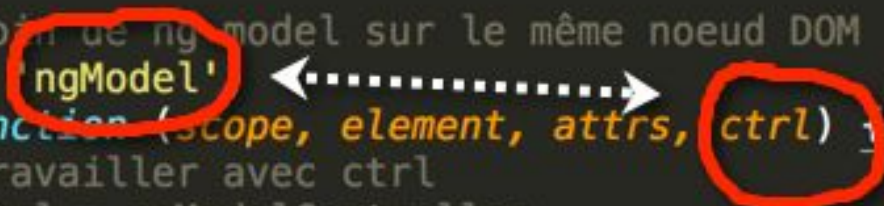
```
<div ng-model="data.user"><input type="text" ma-directive /></div>
```



# Directive - L'option require

- Exemple de directive avec require
  - [http://localhost:8080/day\\_09/step\\_01](http://localhost:8080/day_09/step_01)

```
.directive('asyncEmail', function($q, $timeout) {  
  return {  
    // A besoin de ng model sur le même noeud DOM  
    require: 'ngModel',  
    link: function(scope, element, attrs, ctrl) {  
      // Travailler avec ctrl  
      // ctrl = ngModelController  
    }  
  }  
});
```



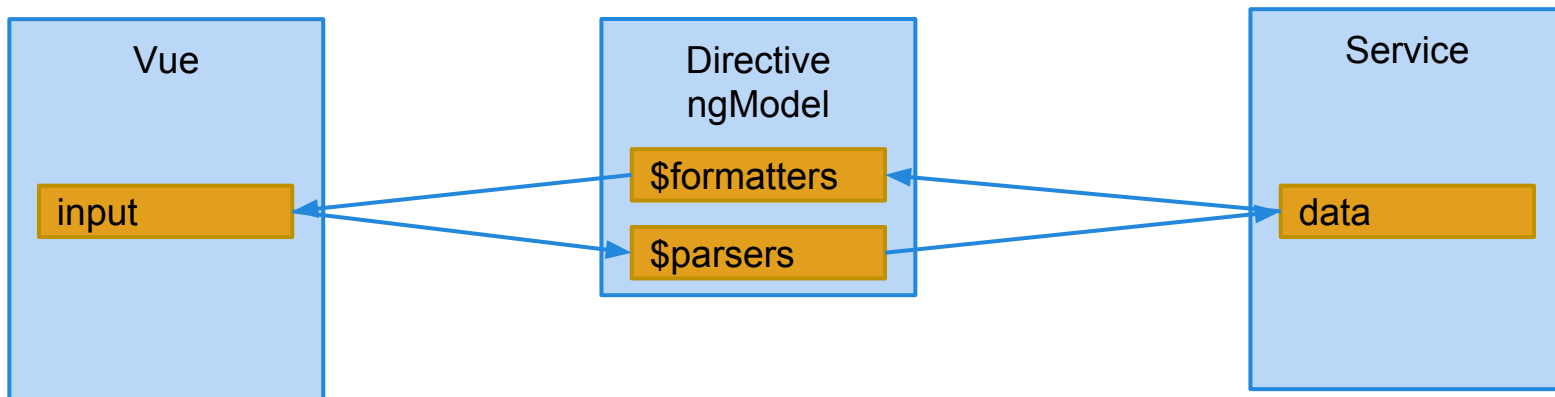
A diagram illustrating the relationship between the `require` and `link` functions in the directive definition. A red circle highlights `ngModel` in the `require` property, and another red circle highlights `ctrl` in the `link` function's arguments. A dashed double-headed arrow connects these two circles, indicating that `ctrl` is the `ngModel` controller.

*ctrl* ⇒ <https://docs.angularjs.org/api/ng/type/ngModel.NgModelController>

# Directive - ngModelController.\$formatters & \$parsers

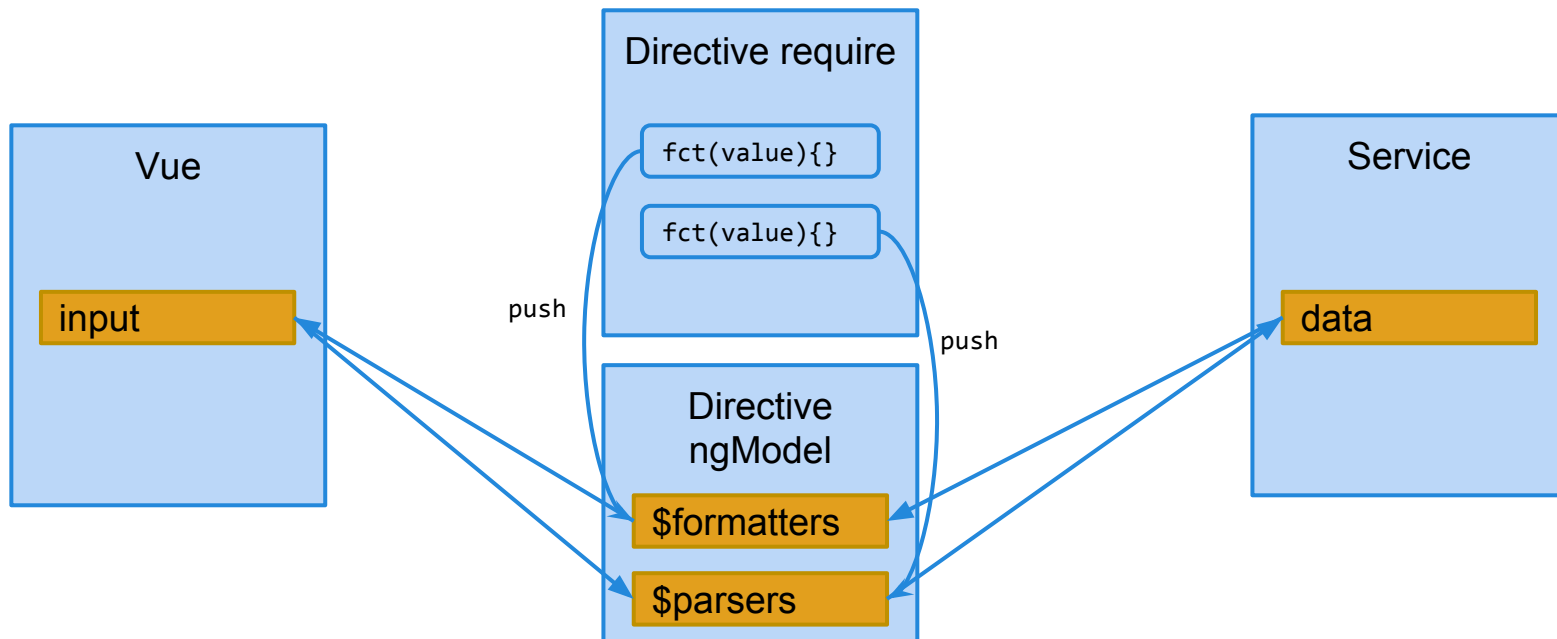
- **\$formatters** (model2view)
- **\$parsers** (view2model)
- Tableau de fonction
  - `function (value) { /*Transformer value en newValue*/ return newValue; }`

## Schéma classique



# Directive - ngModelController.\$formatters & \$parsers

## Schéma avec \$formatters & \$parsers



# Directive - ngModelController.\$formatters & \$parsers

- Example de directive avec \$formatters & \$parsers
  - [http://localhost:8080/day\\_08/step\\_02](http://localhost:8080/day_08/step_02)

```
.directive('myDate', function($q, $timeout) {  
  return {  
    // A besoin de ng-model sur le même noeud DOM  
    require: 'ngModel',  
    link: function (scope, element, attrs, ngModelCtrl) {  
      // Travailler avec ctrl  
      // ctrl = ngModelController  
      ngModelCtrl.$formatters.push(function(value) {  
        return new Date(value);  
      });  
  
      ngModelCtrl.$parsers.push(function(value) {  
        return new Date(value).getTime();  
      });  
    }  
  }  
});
```

# Directive - ngModelController.\$validators

- Example de directive avec \$validators
  - [http://localhost:8080/day\\_09/step\\_03](http://localhost:8080/day_09/step_03)

```
.directive('fullnameValidator', function($q, $timeout) {  
  return {  
    // A besoin de ng-model sur le même noeud DOM  
    require: 'ngModel',  
    link: function (scope, element, attrs, ngModelCtrl) {  
      // Travailler avec ctrl  
      // ngModelCtrl = ngModelController  
      ngModelCtrl.$validators.fullname = function(modelValue, viewValue) {  
        var value = modelValue || viewValue;  
        return /^[a-zA-Z]+ [a-zA-Z]+ <[^>]+>$/.test(value);  
      };  
    }  
  }  
});
```

# Directive - ngModelController.\$asyncValidators

- Exemple de directive avec \$asyncValidators
  - [http://localhost:8080/day\\_08/step\\_04](http://localhost:8080/day_08/step_04)

```
.directive('asyncUsername', function($q, $timeout) {  
  return {  
    // A besoin de ng-model sur le même noeud DOM  
    require: 'ngModel',  
    link: function (scope, element, attrs, ngModelCtrl) {  
      // Travailler avec ctrl  
      // ngModelCtrl = ngModelController  
      ngModelCtrl.$asyncValidators.username = function(modelValue, viewValue) {  
        var value = modelValue || viewValue;  
        var deferred = $q.defer();  
        $timeout(function() {  
          var usernames = ['firehist', 'toto', 'tartoprune'];  
          if (usernames.indexOf(value) > -1) {  
            deferred.reject();  
          } else {  
            deferred.resolve();  
          }  
        }, 2000);  
        return deferred.promise;  
      };  
    }  
  }  
});
```

# Directive - ngModelController

- Aller plus loin !
  - <https://docs.angularjs.org/api/ng/type/ngModel.NgModelController>

# Directive - ngModelOptions

- Permet de configurer le ng-model
  - <https://docs.angularjs.org/api/ng/directive/ngModelOptions#usage>
- Options
  - updateOn, debounce, getterSetter, allowInvalid, timezone
- DEMO
  - [http://localhost:8080/day\\_09/step\\_05](http://localhost:8080/day_09/step_05)



# Directive - ngMessages

- Module externe *ngMessages*
  - <https://docs.angularjs.org/api/ngMessages/directive/ngMessages#usage>
  - Attention : Affiche un seul message à la fois (ordre du DOM)  
Utiliser l'attribut *ng-messages-multiple*  
`<div ng-messages="myForm.myField" ng-messages-multiple>...</div>`
- DEMO
  - [http://localhost:8080/day\\_09/step\\_06](http://localhost:8080/day_09/step_06)

# Formulaire & Directive

- TP
  - BASE : [http://localhost:8080/day\\_09/step\\_07](http://localhost:8080/day_09/step_07)
  - Implémenter ngMessages
  - Implémenter un \$parsers/\$formatters
  - Implémenter un \$validators
  - Implémenter un \$asyncValidators
  - Ecrire du style CSS pour les erreurs !

# 3 - Les principaux outils

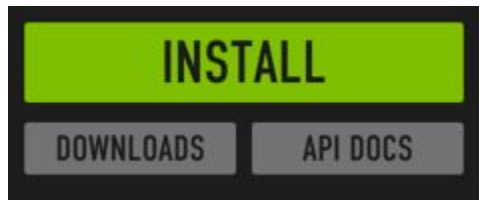


# Outils - Dépendances

- NodeJS (<https://nodejs.org/>)

- Moteur JavaScript

- Installation :  
***click & run***



- NodeJS, mais à quoi ça sert ?

- <http://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js/node-js-mais-a-quoi-ca-sert>

# Outils - Dépendances

- NPM (<http://npmjs.org>)
  - Gestion de dépendances
  - Installation avec NodeJS
- Liens pour en savoir plus !
  - <http://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js/les-modules-node-js-et-npm>



# Outils - Dépendances

- Bower (<http://bower.io>)
  - Gestion de dépendances
  - *npm install -g bower*
- Liens pour en savoir plus !
  - Paquet NPM pour Bower : <https://www.npmjs.com/search?q=bower->
  - <http://www.alsacreations.com/tuto/lire/1609-bower-pour-les-nuls.html>



# Outils - Automatisation

- GruntJS (<http://gruntjs.com>)
  - `npm install -g grunt-cli`
  - Automatise les tâches répétitives  
`grunt taskName:taskOption`
- Liens pour en savoir plus !
  - Paquet NPM pour Grunt : <https://www.npmjs.com/search?q=grunt->
  - [http://www.dailymotion.com/video/x18gtfg\\_tutoriel-grunt-decouverte-de-](http://www.dailymotion.com/video/x18gtfg_tutoriel-grunt-decouverte-de-)



# Outils - Scaffolding

- Yeoman (<http://yeoman.io>)
  - Utilise **Grunt** & **Bower**
  - Generators
    - <http://yeoman.io/generators/>
- Example (appli firebase)

```
npm install -g yo generator-angularfire  
mkdir monprojet && cd $_  
yo angularfire monprojet
```

GitHub : <https://github.com/firebase/generator-angularfire>



→ Build : grunt  
→ Dev : grunt serve



# Outils - Testing Unitaire

- Karma (<http://karma-runner.github.io>)

- Lanceur de test



- Jasmine (<http://jasmine.github.io/>)

- Framework d'écriture de test

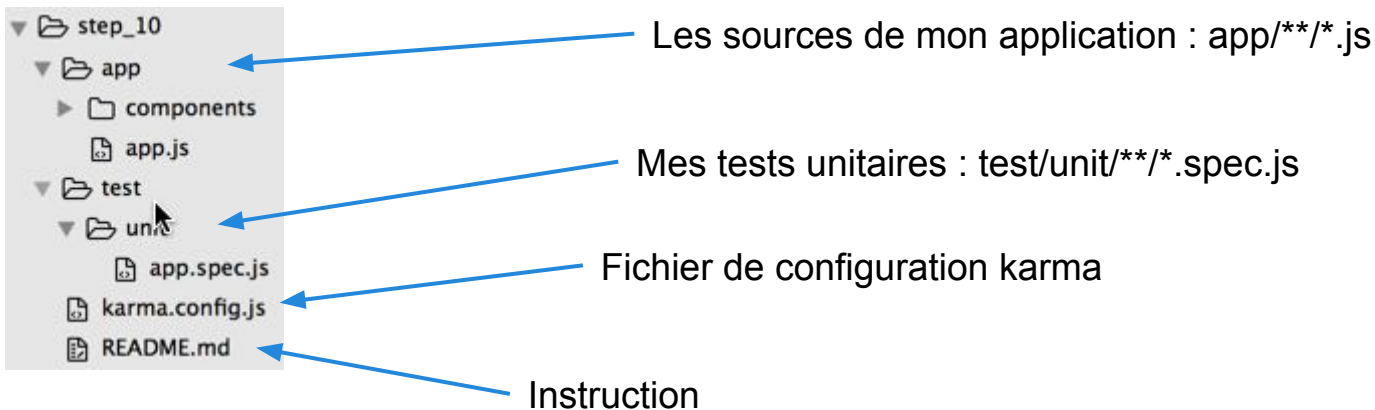


- angular-mocks (<https://docs.angularjs.org/api/ngMock>)

- bower install angular-mocks

# Outils - Testing Unitaire - mise en place

## - Architecture



[https://github.com/firehist/angular-training/blob/master/day\\_08/step\\_10/README.md](https://github.com/firehist/angular-training/blob/master/day_08/step_10/README.md)

# Outils - Testing Unitaire

- DEMO
  - [http://localhost:8080/day\\_09/step\\_10](http://localhost:8080/day_09/step_10)

**4 - Aller plus loin ...  
aller plus loin !**

# Liens - Tutoriels & ressources

- Thinkster (<http://thinkster.io>)
  - <https://thinkster.io/a-better-way-to-learn-angularjs/>
- John Papa styleguide
  - <https://github.com/johnpapa/angular-styleguide>
- AngularJS Tutorial
  - <https://docs.angularjs.org/tutorial>
- egghead.io
  - <https://egghead.io/>
- AngularJS Blog
  - <https://blog.angularjs.org/>

# Liens - Les formulaires

- AngularJS Form Validation
  - <https://scotch.io/tutorials/angularjs-form-validation>
- AngularJS Form
  - <https://docs.angularjs.org/guide/forms>
- HTML5 Form Validation
  - <http://www.the-art-of-web.com/html/html5-form-validation/>

# Liens - Le testing

- Unit testing
  - <http://www.smashingmagazine.com/2014/10/07/introduction-to-unit-testing-in-angularjs/>
- e2e testing
  - <https://docs.angularjs.org/guide/e2e-testing>
- Protractor
  - <https://angular.github.io/protractor/#/>