



AngularJS

Jour 9 - Formulaires & directives avancés

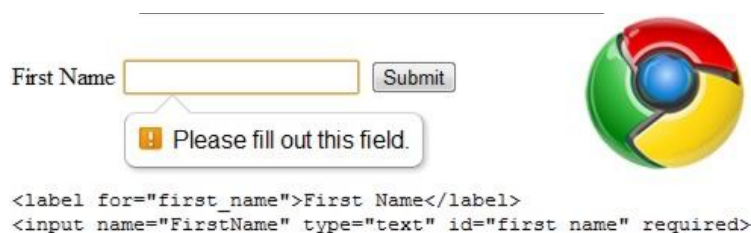
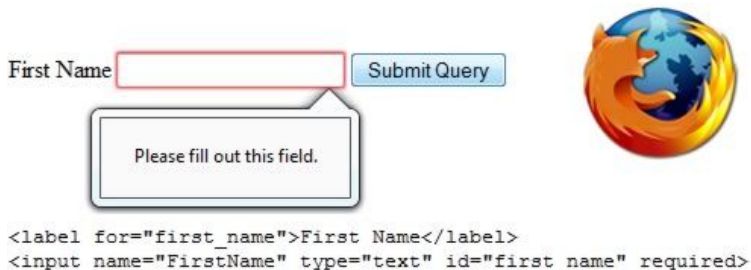
Au programme !

- Les formulaires
- Les directives avancées
- Le testing

1 - Les formulaires

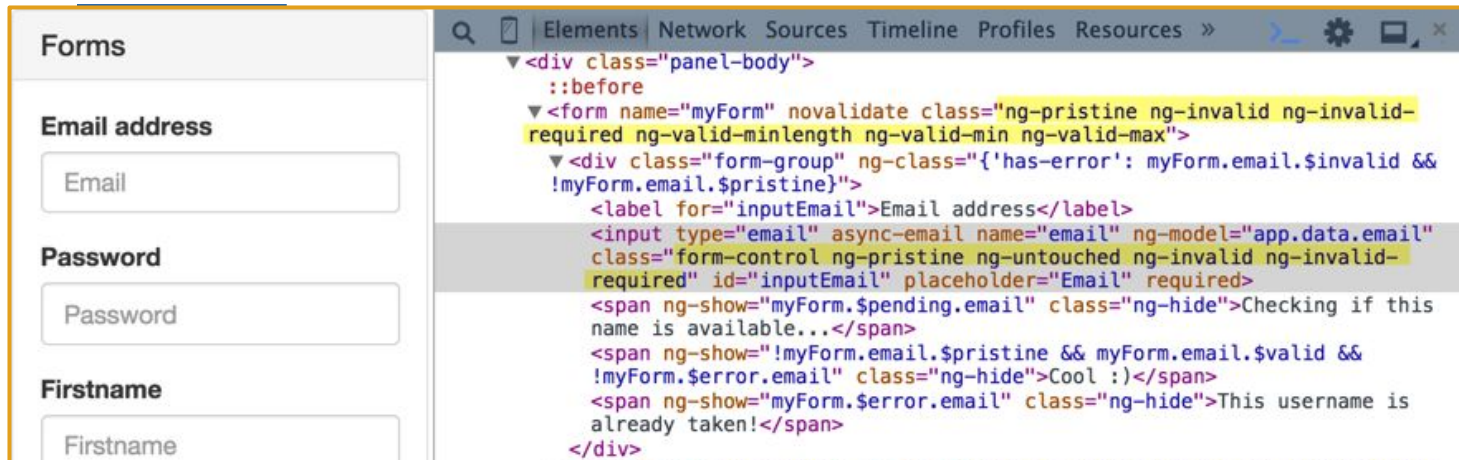
Formulaire - Création d'un formulaire

- 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 screenshot of a web application with a form and its underlying HTML code. The form on the left has three input fields: 'Email address', 'Password', and 'Firstname'. The 'Email address' field is currently active. The code on the right is the HTML for this form, showing various AngularJS classes like 'ng-pristine', 'ng-invalid', and 'ng-invalid-required' applied to the form and its controls. The code also includes validation logic for email availability and uniqueness.

```
<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>
```

Formulaire - L'objet Formulaire

- Le formulaire publie un objet sur le scope
 - 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

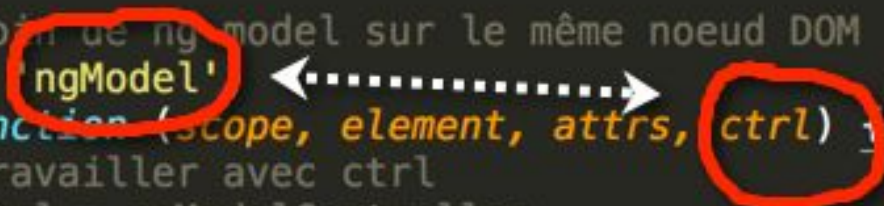
```
<div><input type="text" ma-directive /></div>
```

```
<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

```
.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  
    }  
  }  
});
```



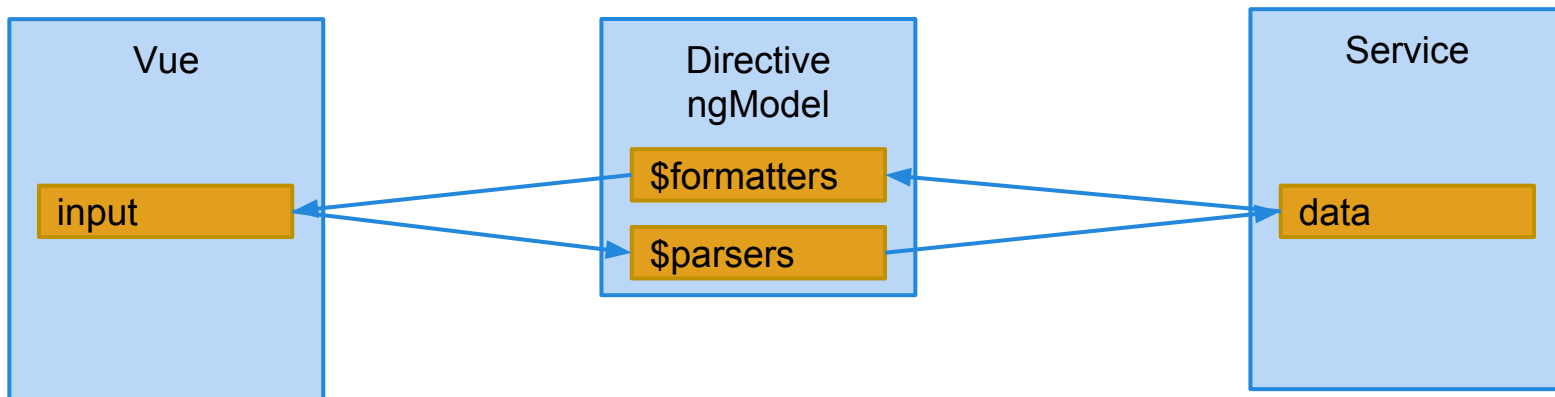
The diagram illustrates the relationship between the `require` and `link` functions in the directive definition. A red circle highlights `ngModel` in the `require` property, and another red circle highlights `ctrl` in the `link` function. A dashed double-headed arrow connects these two elements, indicating that `ctrl` is the `ngModel` controller instance passed to the `link` function.

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

Directive - ngModelController.\$formatters & \$parsers

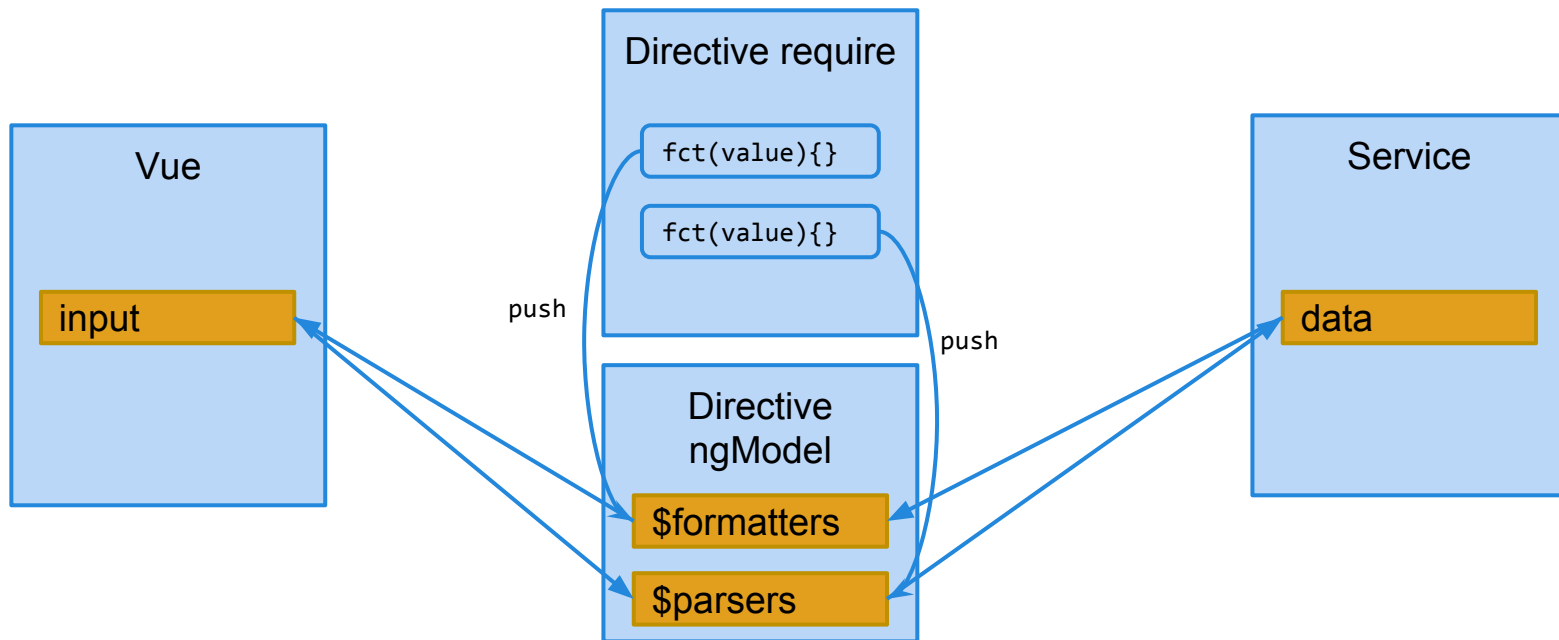
- **\$formatters** (model2view)
- **\$parsers** (view2model)
- Tableau de fonction
 - `function (value) { /*Transformer value en newValue*/ return newValue; }`

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

```
.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

```
.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

```
.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

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

Formulaire & Directive

- TP
 - BASE : 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 !