



FRONT-END FRAMEWORKS

BÀI 6: LÀM VIỆC VỚI FORM VÀ KIỂM LỖI

- ⊙ Biết cách buộc dữ liệu form
- ⊙ Biết cách đổ dữ liệu vào DropDownList
- ⊙ Biết cách kiểm soát dữ liệu form



- 📖 Chỉ thị @ng-model
- 📖 Chỉ thị @ng-model-options
- 📖 Chỉ thị @ng-checked
- 📖 Đổ dữ liệu vào dropdownlist
 - 📖 @ng-repeat
 - 📖 @ng-options
- 📖 Kiểm lỗi form
- 📖 Định dạng trạng thái form



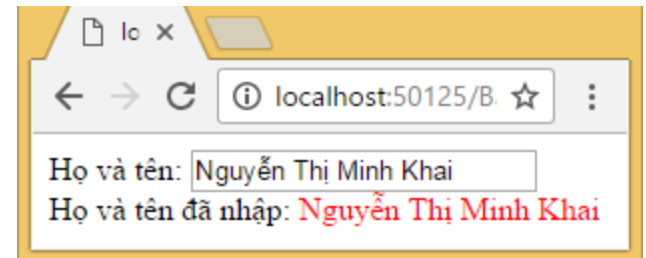
- ❑ @**ng-model** được sử dụng để buộc dữ liệu 2 chiều từ \$scope vào các điều khiển của form như input, select, textarea.
- ❑ Nếu trong \$scope chưa có thuộc tính được buộc lên điều khiển thì thuộc tính đó sẽ tự động được tạo ra
- ❑ Buộc dữ liệu 2 chiều có nghĩa là khi bạn thay đổi dữ liệu trong \$scope thì sẽ làm thay đổi dữ liệu trên các điều khiển và ngược lại.
 - ❖ Ví dụ: <input **ng-model**="name"> sẽ buộc <input> với \$scope.name. Nếu trong \$scope.name đã được định nghĩa thì dữ liệu được hiển thị lên <input>, ngược lại thuộc tính \$scope.name sẽ được tạo ra và sẽ nhận dữ liệu nhập vào <input>.

```
<body ng-app="">
  <form ng-controller="myctrl">
    Họ và tên:
    <input ng-model="name" />
    <br />
    Họ và tên đã nhập:
    <span style="color:red;">{{name}}</span>
    <script>
      function myctrl($scope) {
        $scope.name = "Nguyễn Thị Minh Khai";
      }
    </script>
  </form>
</body>
```

Buộc dữ liệu 2 chiều với \$scope

Hiển thị dữ liệu từ \$scope

Dữ liệu trong \$scope



- ❑ Khi bạn nhập dữ liệu vào ô nhập **họ và tên** sẽ làm thay đổi \$scope.name nên sẽ làm thay đổi **ngay tức thì** dữ liệu hiển thị phần **Họ và tên đã nhập**



Username

Password

Login

EMO



Đọc thông tin của form đăng nhập

- ❑ @ng-model buộc dữ liệu giữa điều khiển và \$scope trong khi đó @ng-model-options qui định cách thức cập nhật dữ liệu từ điều khiển vào \$scope
- ❑ Chú ý: @ng-model-options được giới thiệu từ AngularJS 1.3+
- ❑ Ví dụ sau đây sẽ cập nhật dữ liệu trong \$scope khi điều khiển mất focus

```
<div ng-app="myApp" ng-controller="myCtrl">
  <input ng-model="name" ng-model-options="{updateOn: 'blur'}">
  <div>Đã nhập: {{name}}</div>
</div>
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function ($scope) {
    $scope.name = "Võ Thị Sáu";
  });
</script>
```



Võ Thị Sáu 123|

Đã nhập: Võ Thị Sáu

Con trỏ chưa rời khỏi
điều khiển

Võ Thị Sáu 123

Đã nhập: Võ Thị Sáu 123

Con trỏ đã rời khỏi điều
khiển

Tái hiện slide trước và lý giải thời điểm
thay đổi dòng chữ **Đã nhập**



- ❑ @ng-model-options={updateOn: 'blur'}
 - ❖ Thuộc tính updateOn chỉ ra thời điểm cập nhật dữ liệu trong \$scope
 - ❖ **Blur** là sự kiện xảy ra khi điều khiển mất trở phím. Bạn có thể sử dụng các sự kiện khác như: **click**, **focus**...
- ❑ Ngoài updateOn thì thuộc tính debounce cũng thường được sử dụng để chỉ ra sau bao lâu thì sẽ cập nhật dữ liệu vào \$scope
- ❑ Ví dụ:
 - ❖ ng-model-options={**debounce**: 1000}
 - Sau 1000 mi li giây (1 giây) sẽ cập nhật dữ liệu từ điều khiển vào \$scope

- ❑ Với checkbox thì chỉ thị ng-model buộc trạng thái của checkbox vào \$scope
`<input type="checkbox" ng-model="gender" />`
\$scope.gender sẽ có giá trị true (nếu có chọn) hoặc false nếu không chọn
- ❑ Với radio thì chỉ ng-model buộc giá trị của radio được chọn vào \$scope
`<input type="radio" value="1" ng-model="gender" />`
`<input type="radio" value="2" ng-model="gender" />`
\$scope.gender sẽ có giá trị 1 hoặc 2 tùy thuộc vào radio nào được chọn
- ❑ Chỉ thị **ng-checked**="expr" sẽ chọn radio hay không tùy vào giá trị của biểu thức expr



Username

Password

☐ Remember me?

Login

EMO



Độc thuộc tính checkbox



Username

Password

Remember me?

☐

Yes

☒

No

Login

MO



Độc giá trị radio được chọn

- ❑ Chỉ thị **ng-checked**="expr" sẽ chọn radio/checkbox nếu giá trị của biểu thức expr là true, ngược lại sẽ bỏ chọn

Sở thích : ☐ Tất cả ☐ Đọc sách ☐ Du lịch ☐ Âm nhạc



```
<div class="form-group">
  <label>Sở thích :</label>
  <input type="checkbox" ng-model="all">Tất cả
  <input type="checkbox" ng-model="reading"
    ng-checked="all">Đọc sách
  <input type="checkbox" ng-model="traveling"
    ng-checked="all">Du lịch
  <input type="checkbox" ng-model="music"
    ng-checked="all">Âm nhạc
</div>
```

Country Trung Quốc ▼ OK

- Việt Nam
- Mỹ
- Trung Quốc

- ❑ DropDownList được tạo ra và buộc dữ liệu vào `$scope`
 - ❖ Sử dụng `@ng-model` để buộc giá trị mục chọn vào `$scope`
 - ❖ Sử dụng `@ng-options` hoặc `@ng-repeat` để đổ các mục vào dropdown list.

□ Sử dụng @ng-repeat

```
<div ng-app="">
  <form ng-controller="myctrl">
    <label>Country</label>
    <select ng-model="country">
      <option ng-repeat="x in countries"
              value="{{x}}">{{x}}</option>
    </select>
    <button ng-click="read()">OK</button>
  </form>
</div>
<script>
  function myctrl($scope) {
    $scope.countries = ["Việt Nam", "Mỹ", "Trung Quốc"];
    $scope.read = function () {
      alert($scope.country);
    }
  }
</script>
```



```
<div ng-app="">
  <form ng-controller="myctrl">
    <label>Country</label>
    <select ng-model="country"
      ng-options="x for x in countries"></select>
    <button ng-click="read()">OK</button>
  </form>
</div>
<script>
  function myctrl($scope) {
    $scope.countries = ["Việt Nam", "Mỹ", "Trung Quốc"];
    $scope.read = function () {
      alert($scope.country);
    }
  }
</script>
```

- ❑ @**ng-options**="**label** for **value** in array" tạo danh mục chọn từ các phần tử trong mảng. Mỗi phần tử sẽ tạo một thẻ

`<option value="value">label</option>`

❑ Sự khác nhau giữa @ng-repeat và @ng-options:

- ❖ @ng-repeat value của mỗi option luôn luôn là text
- ❖ @ng-options có thể là đối tượng, mảng...

```
$scope.countries = [  
  { id: "VN", name: "Việt Nam" },  
  { id: "TQ", name: "Trung Quốc" }  
];
```

```
<label>Country</label>  
<select ng-options="x.name for x in countries"  
        ng-model="country"></select>
```

\$scope.country là **object**

```
<label>Country</label>  
<select ng-model="country">  
  <option ng-repeat="x in countries"  
          value="{{x.id}}">{{x.name}}</option>  
</select>
```

\$scope.country là **id**

- ❑ Ngoài ra với **@ng-options** bạn có thể đổ các thuộc tính của một object vào dropdown list

Country

```
<select ng-options="x for (x, y) in country"  
ng-model="prop"></select>
```

```
$scope.country = {  
  id: "VN",  
  name: "Việt Nam",  
  currency: "VNĐ"  
};
```

Ở đây

- X là tên thuộc tính
- Y là giá trị thuộc tính
- ng-model="prop" luôn luôn nhận giá trị thuộc tính được chọn

Country

Trung Quốc ▼
Trung Quốc
Việt Nam

```
<select ng-model="country"  
  ng-options="y.name for (x, y) in countries">  
</select>
```

```
$scope.countries = {  
  VN: {  
    id: "VN",  
    name: "Việt Nam",  
    currency: "VNĐ"  
  },  
  TQ: {  
    id: "TQ",  
    name: "Trung Quốc",  
    currency: "Nhân dân tệ"  
  }  
};
```

**ng-model sẽ buộc giá trị
của mục được chọn, có
nghĩa là một đối tượng**



FRONT-END FRAMEWORKS

BÀI 6 (PHẦN 2)

- ❑ Trong HTML 5 bạn có thể kiểm lỗi form thông qua các thuộc tính của các điều khiển trên form
 - ❖ required
 - ❖ type="email"
 - ❖ ..
- ❑ Các thuộc tính này vẫn chưa phản ánh hết các tình trạng dữ liệu trên form
 - ❖ Dữ liệu đã bị sửa hay chưa
 - ❖ Có tương tác với điều khiển hay chưa
 - ❖ Chưa cung cấp cách định nghĩa thêm các thuộc tính kiểm lỗi mới
 - ❖ ...

- ❑ Với AngularJS các qui luật kiểm lỗi được tăng cường đáng kể, giúp kiểm lỗi thuận tiện hơn.
- ❑ AngularJS không những cho biết tình trạng lỗi của các điều khiển mà còn cho biết tình trạng lỗi của form.
- ❑ Ngoài qui luật kiểm lỗi, AngularJS còn cung cấp các class CSS giúp trình bày lỗi theo từng tình trạng lỗi khác nhau
- ❑ AngularJS cũng cung cấp cách thức để bạn có thể tự định nghĩa thêm các thuộc tính kiểm lỗi riêng của mình.

```
<div ng-app="">
  <form name="frmUser">
    <label>Fullname</label>
    <input name="txtName" ng-model="fullname" required>
    <br />
    <label>Email</label>
    <input name="txtEmail" ng-model="email" type="email">
  </form>
  <h3>Tình trạng lỗi của các điều khiển:</h3>
  <ul>
    <li>Fullname: {{frmUser.txtName.$valid}}</li>
    <li>Email: {{frmUser.txtEmail.$valid}}</li>
  </ul>
  <h3>Tình trạng lỗi của form: {{frmUser.$valid}}</h3>
</div>
```

Fullname

Email

Tình trạng lỗi của các điều khiển:

- Fullname: false
- Email: true

Tình trạng lỗi của form: false

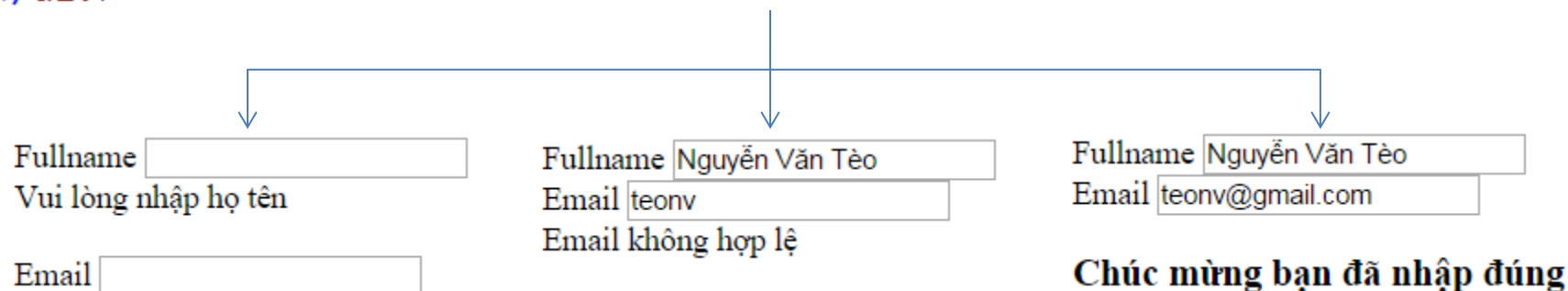
Nhập dữ liệu vào và theo dõi tình trạng của điều khiển và form

- ❑ Để biết được một điều khiển đã nhập đúng như điều kiện kiểm lỗi hay chưa bạn cần kiểm tra \$valid của điều khiển đó:
 - ❖ <tên form>.<tên điều khiển>.\$valid
 - ❖ Ví dụ: frmUser.txtName.\$valid
- ❑ Nếu đã nhập đúng thì giá trị này sẽ là true
- ❑ Form hợp lệ khi tất cả các điều khiển của form hợp lệ
- ❑ Để xác định form đã hợp lệ hay chưa bạn sử dụng thuộc tính \$valid của form
 - ❖ <tên form>.\$valid


```

<div ng-app="">
  <form name="frmUser">
    <label>Fullname</label>
    <input name="txtName" ng-model="fullname" required>
    <div ng-hide="frmUser.txtName.$valid">Vui lòng nhập họ tên</div>
    <br />
    <label>Email</label>
    <input name="txtEmail" ng-model="email" type="email">
    <div ng-hide="frmUser.txtEmail.$valid">Email không hợp lệ</div>
  </form>
  <h3 ng-show="frmUser.$valid">Chúc mừng bạn đã nhập đúng </h3>
</div>

```





DEMO



Tái hiện ví dụ ở slide trước

Thuộc tính	Ý nghĩa	Ví dụ
\$untouched	Chưa tác động	frm1.txt1.\$untouched
\$touched	Đã tác động	frm1.txt1.\$touched
\$pristine	Chưa có sửa chữa	frm1.txt1.\$pristine
\$dirty	Đã có sửa chữa	frm1.txt1.\$dirty
\$invalid	Chưa hợp lệ	frm1.txt1.\$invalid
\$valid	Đã hợp lệ	frm1.txt1.\$valid

```
<span ng-show="frmUser.txtName.$dirty">  
    Bạn đã sửa chữa họ và tên  
</span>
```

Thuộc tính	Ý nghĩa	Ví dụ
\$pristine	Chưa có sửa chữa	frm1.\$pristine
\$dirty	Đã có sửa chữa	frm1.\$dirty
\$invalid	Chưa hợp lệ	frm1.\$invalid
\$valid	Đã hợp lệ	frm1.\$valid
\$submitted	Đã gửi dữ liệu	frm1.\$submitted

```
<span ng-show="frmUser.$pristine">  
    Chưa sửa dữ liệu trên form  
</span>
```

- ❑ AngularJS cung cấp các CSS class cho phép chúng ta định dạng các trạng thái kiểm lỗi của form
- ❑ Bạn chỉ cần override các class này để định nghĩa các định dạng mới cho việc trình bày trạng thái lỗi
- ❑ Ví dụ:
 - ❖ **input.ng-invalid**{
 - định dạng cho các thẻ <input> có dữ liệu không hợp lệ
 - ❖ **form.ng-pristine**{
 - định dạng cho các form chưa có tác động sửa chữa

```
<style>
```

```
  input.ng-invalid {
    background-color: red;
  }
```

CSS cho các thẻ input có lỗi

```
  form.ng-pristine {
    background-color: yellow;
  }
```

CSS cho các form có lỗi

```
</style>
```

```
<div ng-app="">
```

```
  <form name="frmUser">
```

```
    <label>Email</label>
```

```
    <input name="txtEmail" ng-model="email" type="email" required>
```

```
    <span ng-hide="frmUser.txtEmail.$valid">Email không hợp lệ</span>
```

```
  </form>
```

```
  <h3 ng-show="frmUser.$valid">Chúc mừng bạn đã nhập đúng </h3>
```

```
</div>
```

Email Email không hợp lệ

Email

Chúc mừng bạn đã nhập đúng



DEMO



Tái hiện slide trước

Trạng thái điều khiển

Thuộc tính	Ý nghĩa
.ng-untouched	Chưa tác động
.ng-touched	Đã tác động
.ng-pristine	Chưa có sửa chữa
.ng-dirty	Đã có sửa chữa
.ng-invalid	Chưa hợp lệ
.ng-valid	Đã hợp lệ
.ng-invalid- key	Chưa hợp lệ key
.ng-valid- key	Đã hợp lệ key

```
input.ng-invalid{  
    background: gray;  
}
```

Trạng thái form

Thuộc tính	Ý nghĩa
.ng-pristine	Chưa có sửa chữa
.ng-dirty	Đã có sửa chữa
.ng-invalid	Chưa hợp lệ
.ng-valid	Đã hợp lệ
.ng-submitted	Đã gửi dữ liệu
.ng-invalid- key	Chưa hợp lệ key
.ng-valid- key	Đã hợp lệ key

```
form.ng-invalid{  
    background: pink;  
}
```


- ❑ AngularJS cho phép định nghĩa thêm các thuộc tính kiểm lỗi mới.
- ❑ Sau đây là cấu trúc mã định nghĩa chỉ thị kiểm lỗi mới có tên là @even-number.

```
<div ng-app="myapp">
  <form name="frmUser">
    <input name="txtAge" ng-model="age" even-number>
    <span ng-show="frmUser.txtAge.$invalid">Vui lòng nhập số chẵn</span>
  </form>

  <script>
    var app = angular.module('myapp', []);
    app.directive('evenNumber', function () {...});
  </script>
</div>
```

Mã kiểm lỗi được
viết ở đây

❑ B1: Tạo đối tượng ứng dụng

❖ `var app = angular.module('myapp', []);`

❑ B2: Định nghĩa chỉ thị

❖ `app.directive('evenNumber', function () {...});`

❑ B3: Sử dụng chỉ thị

❖ `<input ng-model="age" even-number>`

❑ Chú ý:

❖ Tên định nghĩa trong JavaScript là evenNumber (theo qui ước **camel**)

❖ Tên sử dụng trên các thẻ là even-number (mỗi từ cách nhau dấu -)

```
app.directive('evenNumber', function () {  
    return {  
        require: 'ngModel',  
        link: function (scope, element, attr, mCtrl) {  
            function fnValidate(value) {  
                if (parseInt(value) % 2 == 0) {  
                    mCtrl.$setValidity('charE', true);  
                } else {  
                    mCtrl.$setValidity('charE', false);  
                }  
                return value;  
            }  
            mCtrl.$parsers.push(fnValidate);  
        }  
    };  
});
```

❑ Hàm xử lý chỉ thị phải return một đối tượng gồm 2 thuộc tính là **require** và **link**

❖ **require:** 'ngModel'

❖ **link:** function(scope, element, attr, mCtrl). Trong hàm này chứa hàm fnValidate(value), mã kiểm tra lỗi được viết ở đây để kiểm tra lỗi đối số value.



DEMO



Kiểm lỗi nhập số nguyên tố

- ☑ Chỉ thị @ng-model
- ☑ Chỉ thị @ng-model-options
- ☑ Chỉ thị @ng-checked
- ☑ Đổ dữ liệu vào dropdownlist
 - ☑ @ng-repeat
 - ☑ @ng-options
- ☑ Kiểm lỗi form
- ☑ Định dạng trạng thái form





Cảm ơn