AngularJS Form Validation

[« Previous](http://www.w3schools.com/angular/angular_forms.asp)

[Next Chapter »](http://www.w3schools.com/angular/angular_api.asp)

AngularJS can validate input data.

Form Validation

AngularJS offers client-side form validation.

AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

AngularJS also holds information about whether they have been touched, or modified, or not.

You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

|  |  |
| --- | --- |
| **Note** | Client-side validation cannot alone secure user input. Server side validation is also necessary. |

Required

Use the HTML5 attribute required to specify that the input field must be filled out:

Example

The input field is required:

<form name="myForm">  
<input name="myInput" ng-model="myInput" required>  
</form>  
  
<p>The input's valid state is:</p>  
<h1>{{myForm.myInput.$valid}}</h1>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_required)

E-mail

Use the HTML5 type email to specify that the value must be an e-mail:

Example

The input field has to be an e-mail:

<form name="myForm">  
<input name="myInput" ng-model="myInput" type="email">  
</form>  
  
<p>The input's valid state is:</p>  
<h1>{{myForm.myInput.$valid}}</h1>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_email)

Form State and Input State

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

* $untouched The field has not been touched yet
* $touched The field has been touched
* $pristine The field has not been modified yet
* $dirty The field has been modified
* $invalid The field content is not valid
* $valid The field content is valid

They are all properties of the input field, and are either true or false.

Forms have the following states:

* $pristine No fields have been modified yet
* $dirty One or more have been modified
* $invalid The form content is not valid
* $valid The form content is valid
* $submitted The form is submitted

They are all properties of the form, and are either true or false.

You can use these states to show meaningful messages to the user. Example, if a field is required, and the user leaves it blank, you should give the user a warning:

Example

Show an error message if the field has been touched AND is empty:

<input name="myName" ng-model="myName" required>  
<span ng-show="myForm.myName.$touched && myForm.myName.$invalid">The name is required.</span>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_show)

CSS Classes

AngularJS adds CSS classes to forms and input fields depending on their states.

The following classes are added to, or removed from, input fields:

* ng-untouched The field has not been touched yet
* ng-touched The field has been touched
* ng-pristine The field has not been  modified yet
* ng-dirty The field has been modified
* ng-valid The field content is valid
* ng-invalid The field content is not valid
* ng-valid-*key* One *key* for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-*key* Example: ng-invalid-required

The following classes are added to, or removed from, forms:

* ng-pristine No fields has not been modified yet
* ng-dirty One or more fields has been modified
* ng-valid The form content is valid
* ng-invalid The form content is not valid
* ng-valid-*key* One *key* for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-*key* Example: ng-invalid-required

The classes are removed if the value they represent is false.

Add styles for these classes to give your application a better and more intuitive user interface.

Example

Apply styles, using standard CSS:

<style>

input.ng-invalid {  
    background-color: pink;  
}  
input.ng-valid {  
    background-color: lightgreen;  
}

</style>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_classes)

Forms can also be styled:

Example

Apply styles for unmodified (pristine) forms, and for modified forms:

<style>

form.ng-pristine {  
    background-color: lightblue;  
}  
form.ng-dirty {  
    background-color: pink;  
}

</style>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_classes_form)

Custom Validation

To create your own validation function is a bit more tricky. You have to add a new directive to your application, and deal with the validation inside a function with certain specified arguments.

Example

Create your own directive, containing a custom validation function, and refer to it by using my-directive.

The field will only be valid if the value contains the character "e":

<form name="myForm">  
<input name="myInput" ng-model="myInput" required my-directive>  
</form>  
  
<script>

var app = angular.module('myApp', []);  
app.directive('myDirective', function() {  
  return {  
    require: 'ngModel',  
    link: function(scope, element, attr, mCtrl) {  
      function myValidation(value) {  
        if (value.indexOf("e") > -1) {  
          mCtrl.$setValidity('charE', true);  
        } else {  
          mCtrl.$setValidity('charE', false);  
        }  
        return value;  
      }  
      mCtrl.$parsers.push(myValidation);  
    }  
  };  
});

</script>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_custom)

Example Explained:

In HTML, the new directive will be referred to by using the attribute my-directive.

In the JavaScript we start by adding a new directive named myDirective.

Remember, when naming a directive, you must use a camel case name, myDirective, but when invoking it, you must use - separated name, my-directive.

Then, return an object where you specify that we require  ngModel, which is the ngModelController.

Make a linking function which takes some arguments, where the fourth argument, mCtrl, is the ngModelController,

Then specify a function, in this case named myValidation, which takes one argument, this argument is the value of the input element.

Test if the value contains the letter "e", and set the validity of the model controller to either true or false.

At last, mCtrl.$parsers.push(myValidation); will add the myValidation function to an array of other functions, which will be executed every time the input value changes.

Validation Example

<!DOCTYPE html>  
<html>  
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>  
<body>  
  
<h2>Validation Example</h2>  
  
<form  ng-app="myApp"  ng-controller="validateCtrl"  
name="myForm" novalidate>  
  
<p>Username:<br>  
  <input type="text" name="user" ng-model="user" required>  
  <span style="color:red" ng-show="myForm.user.$dirty && myForm.user.$invalid">  
  <span ng-show="myForm.user.$error.required">Username is required.</span>  
  </span>  
</p>  
  
<p>Email:<br>  
  <input type="email" name="email" ng-model="email" required>  
  <span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid">  
  <span ng-show="myForm.email.$error.required">Email is required.</span>  
  <span ng-show="myForm.email.$error.email">Invalid email address.</span>  
  </span>  
</p>  
  
<p>  
  <input type="submit"  
  ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||  
  myForm.email.$dirty && myForm.email.$invalid">  
</p>  
  
</form>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('validateCtrl', function($scope) {  
    $scope.user = 'John Doe';  
    $scope.email = 'john.doe@gmail.com';  
});  
</script>  
  
</body>  
</html>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate)

|  |  |
| --- | --- |
| **Note** | The HTML form attribute **novalidate** is used to disable default browser validation. |

Example Explained

The AngularJS directive **ng-model** binds the input elements to the model.

The model object has two properties: **user** and **email**.

Because of **ng-show**, the spans with color:red are displayed only when user or email is **$dirty** and **$invalid**.

[« Previous](http://www.w3schools.com/angular/angular_forms.asp)

[Next Chapter](http://www.w3schools.com/angular/angular_api.asp)