**18AIC302J**

**WEEK 7**

**AngularJS Form Validation**

A form is made up of various elements, such as checkboxes, input textboxes, number text, and email text. AngularJS allows us to validate form and input field controls on the client side rather than compared to server-side validations. AngularJS inform users of invalid data while submitting forms with a better user experience.

There are three states that are used to detect the errors:

$dirty: This says that the value has been altered.

$error: This displays the exact error.

$invalid: This states that entered value is invalid.

**States of AngularJS Form Validation**

AngularJS stores the data of changes in the input field. Let's have a look at the states of form input fields.

|  |  |
| --- | --- |
| **States** | **Description** |
| **$valid** | It indicates that the field's content is valid. |
| **$invalid** | It indicates that the field's content is not valid. |
| **$dirty** | It shows that the field is modified. |
| **$pristine** | It shows that the field is not modified yet. |
| **$touched** | It indicates that the field is touched. |
| **$untouched** | It indicates that the field is not touched. |
| **$submitted** | It indicates that the form has been submitted. |

**Validation Directives**

Now, let's see the validation directives in the AngularJS Form Validation.

|  |  |
| --- | --- |
| **Directive** | **Description** |
| **ng-required** | It is used to make a field required to fill. |
| **ng-minlength** | It is used to set a minimum length of the input. |
| **ng-maxlength** | It is used to set the maximum length of the input. |
| **ng-pattern** | It shows a pattern validation error key if the value of the ngModel parameter does not match the given RegEx expression. |

### Example 1

We are creating a simple input box to send a validated text to the user. Let's have a look.

<!DOCTYPE html>

<html>

   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

   <body ng-app="">

       <p>Enter your name in the input box:</p>

       <form name="myForm">

           <input name="myInput" ng-model="myInput" required />

       </form>

       <h3>Validation State is:</h3>

       <h1>{{myForm.myInput.$valid}}</h1>

   </body>

</html>

**Output:**

**Explanation**:

* The validation state will show the message "false" if the input box is empty.
* Similarly, the validation state will show the message "true" as soon as we enter something in the input box.

### Example 2

In our next example, we will create a form asking for a username and age. Let's check it out.

<!DOCTYPE html>

<html>

   <head>

       <title>AngularJs Form Validation</title>

       <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

       <script type="text/javascript">

           var app = angular.module('formApp', []);

           app.controller('formCtrl', function ($scope) {

               $scope.sendForm = function () {

                   $scope.msg = 'Form Validated Successfully!';

               };

           });

       </script>

   </head>

   <body>

       <div ng-app="formApp" ng-controller="formCtrl">

           <form name="personForm" ng-submit="sendForm()">

               Name: <input type="text" name="name" ng-model="person.name" required />

               <span ng-show="personForm.name.$error.required"> Required! </span>

               <br /><br />

               Age: <input type="number" name="age" ng-model="person.age" required />

               <span ng-show="personForm.age.$error.required"> Required! </span>

               <br /><br />

               <button type="submit">Submit</button><br /><br />

               <span>{{msg}}</span>

           </form>

       </div>

   </body>

</html>

**Output:**

**Explanation**:

Let's understand the output one by one.

At first, the form will ask for the Name and the Age. It has a submit button also.

Now the question is, Will our form still validate if we did not fill or misses any field? The answer is "No". The form will ask you to fill out the fields you missed.

The message "Form Validated Successfully!" will be displayed when the details are entered correctly.

In this example, the first field (name) will accept any text, but it will only take a number in the second field (age).

## Custom Validation

Custom Validation is a term used while creating conditions that are not declared by default. For instance, you can set your validations at any specific place that will be checked while validating any result.

AngularJS offers HTML input types like text, number, date, radio, checkbox, etc., and validation directives like min, max, pattern, required, etc., to execute custom validation. We can make our validations using the $validators. The $validators accept two parameters, modelValue and viewValue. A $setValidity is called internally to return a true or false value. This process repeats itself with every change in the input.  
  
In the below example, the $parsers command takes myValidation as a parameter. An expression can be parsed and evaluated using the function that the $parsers returns.

Let's take a quick example of this.

<!DOCTYPE html>

<html>

   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

   <body ng-app="myApp">

       <p>Fill the given form:</p>

       <form name="myForm">

           Name:

           <input

               name="myInput"

               ng-model="myInput"

               required

               my-directive

           /><br /><br />

           Age:

           <input name="age" type="number" required /><br /><br />

       </form>

       <p>

           The Name field must contain the character "N" to be considered as valid.

       </p>

       <h3>Validation State is:</h3>

       <h1>{{myForm.myInput.$valid}}</h1>

       <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('N') > -1) {

                               mCtrl.$setValidity('charE', true);

                           } else {

                               mCtrl.$setValidity('charE', false);

                           }

                           return value;

                       }

                       mCtrl.$parsers.push(myValidation);

                   },

               };

           });

       </script>

   </body>

</html>

**Output:**

**Explanation**:

The output shows the empty fields with a false validation state. This will come to your screen when you first run the program.

The validation state is true in the output as the Name field has the character ‘N’ in it.

The validation state will remain false until it encounters the character ‘N’ in the Name field.