17CS2015 Web Technology Lab

Ex. 10. Form validation using ANGULARJS

URK17CS127

Date:

To design a single page web application using HTML5 and validate the user inputs using AngularJS.

Part 1: HTML inputs

1. Required

AngularJS provides a **ng-required** directive have the flexibility to set the input field should have a value or not. The following syntax is used to make sure that the input field should not be empty. We can set it to false if you do not want to restrict this.

```
<input type="text" ng-required="true" />
```

2. Minimum Length

The directive **ng-minlength** is used to validate the minimum length of the input value. This will make sure that the length of the entered value not less than the value set using ng-minlength directive.

```
<input type="text" ng-minlength=10>
```

3. Maximum Length

The directive **ng-maxlength** is used to validate the maximum length of the input value. This will make sure that the length of the entered value is not more than the value set using ng-max length directive.

```
<input type="text" ng-maxlength=20 />
```

4. Pattern

The **ng-pattern** directive is used to ensure that an input matches a regex pattern, the following syntax is used.

```
<input type="text" ng-pattern="[a-zA-Z]">
```

5. Email

We can set the input type to email to ensure that the input field is a valid email id.

```
<input type="email" name="email" ng-model="user.email">
```

6. Number

We can set the input type to number to ensure that the input field is a number.

```
<input type="number" name="age" ng-model="user.age">
```

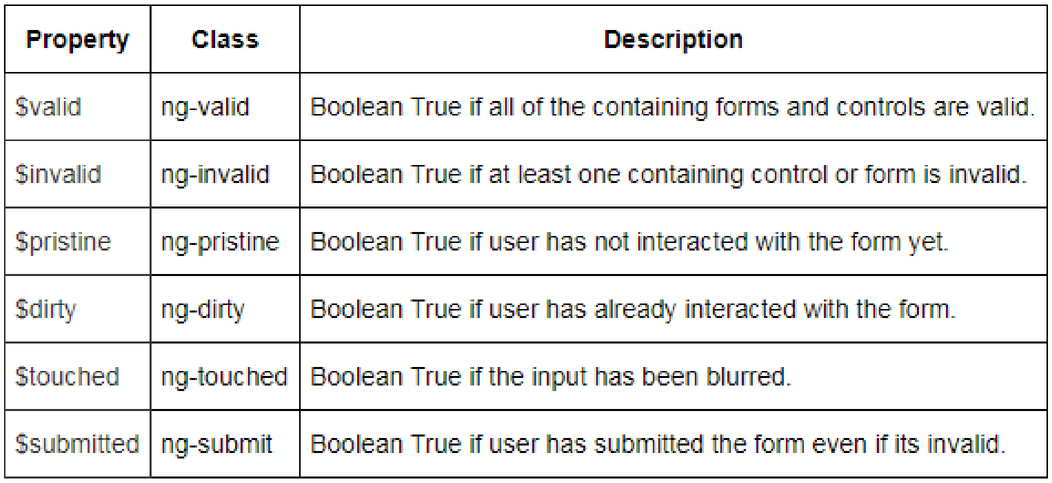
7. URL

We can set the input type to url to ensure that the input field is a url.

```
<input type="url" name="homepage" ng-model="user.url">
```

Part 2: Angular Form Validation Properties

Angular provides properties on forms to keep track of all its controls and nested forms as well as the state of them, such as being valid/invalid or dirty/pristine. The following table describes those properties and corresponding angular classes that help us to validate forms.



Property	Class	Description
$valid	ng-valid	Boolean True if all of the containing forms and controls are valid.
$invalid	ng-invalid	Boolean True if at least one containing control or form is invalid.
$pristine	ng-pristine	Boolean True if user has not interacted with the form yet.
$dirty	ng-dirty	Boolean True if user has already interacted with the form.
$touched	ng-touched	Boolean True if the input has been blurred.
$submitted	ng-submit	Boolean True if user has submitted the form even if its invalid.

Part 3: Angular Form Validation Using CSS Classes

While handling forms angularJS adds specific classes to the form based upon their state. To allow styling of form as well as controls, ngModel adds these CSS classes:

- ng-valid
- ng-invalid
- ng-pristine
- ng-dirty
- ng-touched
- ng-untouched

Sample Code

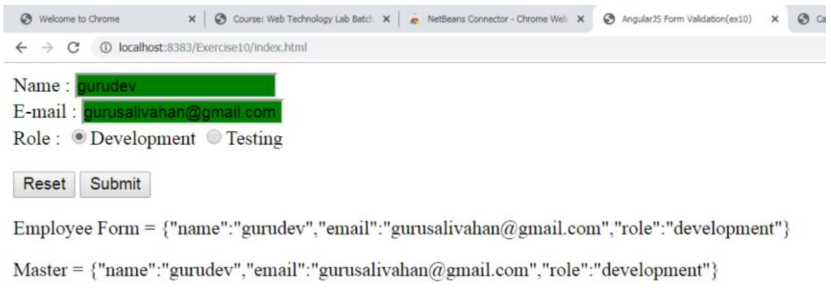
```
<!doctype html>

<html>
```

```
	<head>
		<meta charset="UTF-8">
		<title>AngularJS Form Validation</title>
		<script type="text/javascript" src="angular.min.js"></script>
		<style type="text/css">
				.f1 input.ng-invalid.ng-touched
				{
				background-color:red;
				}
				.f1 input.ng-valid.ng-touched
				{
				background-color:green;
				}
		</style>
	</head>
<body ng-app="formApp" ng-controller="fctrl">
<form name="form" novalidate class="f1">
Name : <input type="text" ng-model="employee.name" required><br>
E-mail : <input type="email" ng-model="employee.email" required><br>
Role : <input type="radio" ng-model="employee.role" value="development">Development
<input type="radio" ng-model="employee.role" value="testing">Testing<br><br>
<input type="button" ng-click="reset()" value="Reset">
<input type="submit" ng-disabled="form.$invalid" ng-click="save(employee)" value="Submit"
/>
</form>
  <p>Employee Form = {{employee}}</p>
  <p>Master = {{master}}</p>
   <script>
		var app = angular.module('formApp',[]);
```

```
                app.controller('fctrl', function($scope){
                $scope.master = {};
                $scope.save= function(employee) {
                        $scope.master = angular.copy(employee);
                }
                $scope.reset = function() {
                        $scope.employee = angular.copy($scope.master);
                }
        });
        </script>
        </body>
</html>
```

Sample Valid Page Output



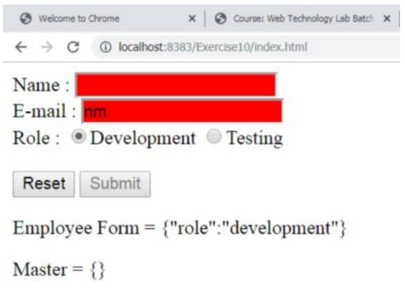
```
Name : gurudev
E-mail : gurusalivahan@gmail.com
Role : Development  Testing

Reset  Submit

Employee Form = {"name":"gurudev","email":"gurusalivahan@gmail.com","role":"development"}

Master = {"name":"gurudev","email":"gurusalivahan@gmail.com","role":"development"}
```

Sample Invalid Page Output



```
Name :
E-mail : nm
Role : Development  Testing

Reset  Submit

Employee Form = {"role":"development"}

Master = {}
```

Web site URL: http://urk17cs127.rf.gd/Exercise10/

You tube URL: **https://youtu.be/QAT-NvxhD7w**