**Angular JS – Day 3 (Form Handling)**

**Exercises**

1. Create an Angular JS form with a text field and a button with the label “Greet”. After entering a name and click on Greet button, it should display alert message “Hello <<name>>”

Define a controller GreetController and attach to <body>

Create a function wish() inside the GreetController and display the alert message from this function wish().

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Two</title>  
 <script src="angular.min.js"></script>  
 </head>  
 <form>  
 <body ng-app="" ng-controller="GreetController">  
 <input type="text" ng-model="name">  
 <button ng-click="wish(name)">Greet</button>  
 </form>  
 </body>  
   
   
 <script>  
 **function** GreetController($scope)  
 **{**  
 $scope.wish=**function**(name)  
 **{**  
 **alert**('Hello '+name);  
 **}**  
   
 **}**  
 </script>  
</html>

1. In the above example, provide a reset button and upon clicking the reset button the value in the text field should be cleared.

Note: There are two ways

Way 1: You could create reset button with <input type=”reset”>

Way 2: You could create a function with the name reset() inside the Controller and provide a simple button, upon clicking this button, you could call the reset() method from the controller.

Hint:

<button ng-click=”reset()”>Reset</button>

reset() is a function in the controller.

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Two</title>  
 <script src="angular.min.js"></script>  
 </head>  
 <form>  
 <body ng-app="" ng-controller="GreetController">  
 <input type="text" ng-model="name">  
 <button ng-click="wish(name)">Greet</button>  
 <input type="reset"/>  
 <button ng-click="reset()">Reset</button>  
 </form>  
 </body>  
   
   
 <script>  
 **function** GreetController($scope)  
 **{**  
 $scope.wish=**function**(name)  
 **{**  
 **alert**('Hello '+name);  
 **}**  
   
 $scope.reset=**function**()  
 **{**  
 $scope.name=*""*;  
 **}**  
   
 **}**  
 </script>  
</html>

3. Rewrite the above program by defining the controller “GreetController” inside a module “GreetApp”.

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Three</title>  
 <script src="angular.min.js"></script>  
 </head>  
  
 <body ng-app="GreetApp" ng-controller="GreetController">  
   
   
 Enter a name: <input type="text" ng-model="name"><br>  
 <button ng-click="wish(name)">Greet</button>  
 <button ng-click="reset()">Reset</button>  
   
 </body>  
   
   
 <script>  
   
 **var** app=angular.module(*"GreetApp"*,[]);  
 app.controller(*"GreetController"*,**function**($scope)   
 **{**  
  
 $scope.wish=**function**(name)  
 **{**  
 **alert**('Hello '+name);  
 **}**  
   
 $scope.reset=**function**()  
 **{**  
 $scope.name=*""*;  
 **}**  
  
 **}**);   
  
 </script>  
</html>

4. Write a simple AngularJS to demonstrate ng-show directive.

Display a checkbox with the label “Show or Hide”.

If the checkbox is clicked/selected, then show the String “Hello AngularJS Experts” in red text. If it is deselected, hide this string.

5. Demonstrate binding the input type “email” and “number” to <h1> tags.

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Email example</title>  
 <script src="angular.min.js"></script>  
 </head>   
   
 <body ng-app="">  
 Enter your email id: <input type="email" ng-model="userEmail"><br>  
 Enter your no id: <input type="number" ng-model="userNumber"><br>  
 <h1 ng-bind="userEmail"></h1>  
 <h1 ng-bind="userNumber"></h1>   
 <body>  
  
 </body>  
</html>

5. Demonstrate two-way data binding in AngularJS.

<!doctype html>  
<html lang="en">  
<head>  
 <title>Two way form handling</title>  
 <script src="angular.min.js"></script>  
 </head>  
<body ng-app="formExample">  
 <div ng-controller="TwoWayController">  
 <form novalidate>  
 Name: <input type="text" ng-model="user.name" /><br />  
 E-mail: <input type="email" ng-model="user.email" /><br />  
 Gender: <input type="radio" ng-model="user.gender" value="male" />male  
 <input type="radio" ng-model="user.gender" value="female" />female<br />  
 <input type="button" ng-click="reset()" value="Reset" />  
 <input type="submit" ng-click="update(user)" value="Save" />  
 </form>  
 <pre>form = {{user|json}}</pre>  
 <pre>master = {{master|json}}</pre>  
</div>  
  
<script>  
 angular.module('formExample', [])  
 .controller('TwoWayController', ['$scope', **function**($scope) **{**  
 $scope.master = **{}**;  
  
 $scope.update = **function**(user) **{**  
 $scope.master = angular.copy(user);  
 **}**;  
  
 $scope.reset = **function**() **{**  
 $scope.user = angular.copy($scope.master);  
 **}**;  
  
 $scope.reset();  
 **}**]);  
</script>  
</body>  
</html>

5. Demonstrate a simple AngularJS form with $error, $dirty and $invalid properties.

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Form properties</title>  
 <script src="angular.min.js"></script>   
 </head>  
 <body ng-app="">  
<form name="MyForm" novalidate>  
   
Enter Employee id:  
<input name="empName" type="text" ng-model="empName" required>  
 <span style="color:red">  
 <span ng-show="MyForm.empName.$error.required">Employee Name is required.</span>  
 </span>  
<br>  
  
<button   
 ng-disabled="MyForm.empName.$dirty && MyForm.empName.$invalid">Submit  
</button>  
  
  
 </body>  
</html>

6. Demonstrate form handling in AngularJS with employee id and email with form validation.

<html>  
<head>  
<title>Angular JS Forms</title>  
</head>  
<body ng-app="">  
<h2>AngularJS Sample Application</h2>  
  
<div>  
  
<form name="employeeForm" novalidate>  
   
Enter Employee id:  
<input name="empid" type="number" ng-model="empId" required>  
 <span style="color:red">  
 <span ng-show="employeeForm.empid.$error.required">Employee id is required.</span>  
 </span>  
<br>  
  
Email:  
<input name="email" type="email" ng-model="email" required>  
<span style="color:red" ng-show="employeeForm.email.$dirty && employeeForm.email.$invalid">  
 <span ng-show="employeeForm.email.$error.required">Email is required.</span>  
 <span ng-show="employeeForm.email.$error.email">Invalid email address.</span>  
 </span>  
<br>  
  
  
  
<button   
 ng-disabled="employeeForm.empid.$dirty && employeeForm.empid.$invalid || employeeForm.empName.$dirty && employeeForm.empName.$invalid"  
 ng-click="submit()">Submit  
</button>  
  
</form>  
</div>  
  
</body>  
<script src="angular.min.js"></script>  
</html>

7. Updating on loss of focus (update on blur):

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>AngularJS update on blur</title>  
 <script src="angular.min.js"></script>   
 </head>  
 <body ng-app="">  
 <div ng-controller="ExampleController">  
 <form>  
 Name:  
 <input type="text" ng-model="my.name" ng-model-options="{updateOn:'blur'}" /><br />  
 City:  
 <input type="text" ng-model="my.city" /><br />   
 </form>  
<pre>username = "{{my.name}}"</pre>  
 <pre>usercity= "{{my.city}}"</pre></div>  
  
 </body>  
 <script>  
 **function** ExampleController($scope)  
 **{**  
 **alert**('controller');  
 $scope.my=**{}**;  
   
 **}**  
 </script>  
</html>

8. Delayed / Debounced update:

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Debounce update</title>  
 <script src="angular.min.js"></script>  
 </head>  
 <body ng-app="">  
<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>AngularJS update on blur</title>  
 <script src="angular.min.js"></script>   
 </head>  
 <body ng-app="">  
<div>  
 <form>  
 Name:  
 <input type="text" ng-model="user.name" ng-model-options="{ debounce: 250 }" /><br />  
 </form>  
 <pre>username = "{{user.name}}"</pre>  
</div>  
 </body>  
</html>

9. Custom validators:

Test.html:

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <title>Custom validations</title>  
 <script src="angular.min.js"></script>  
 <script src="script.js"></script>   
 </head>  
 <body ng-app="form-example1">  
<form name="form"novalidate>  
 <div>  
 Size (integer 0 - 10):  
 <input type="number" ng-model="size" name="size"  
 min="0" max="10" integer />{{size}}<br />  
 <span ng-show="form.size.$error.integer">The value is not a valid integer!</span>  
 <span ng-show="form.size.$error.min || form.size.$error.max">  
 The value must be in range 0 to 10!</span>  
 </div>  
  
 <div>  
 Username:  
 <input type="text" ng-model="name" name="name" username />{{name}}<br />  
 <span ng-show="form.name.$pending.username">Checking if this name is available...</span>  
 <span ng-show="form.name.$error.username">This username is already taken!</span>  
 </div>  
  
</form>  
 </body>  
</html>

script.js:

var app = angular.module('form-example1', []);  
  
var INTEGER\_REGEXP = /^\-?\d+$/;  
app.directive('integer', function() {  
 return {  
 require: 'ngModel',  
 link: function(scope, elm, attrs, ctrl) {  
 ctrl.$validators.integer = function(modelValue, viewValue) {  
 if (ctrl.$isEmpty(modelValue)) {  
 // consider empty models to be valid  
 return true;  
 }  
  
 if (INTEGER\_REGEXP.test(viewValue)) {  
 // it is valid  
 return true;  
 }  
  
 // it is invalid  
 return false;  
 };  
 }  
 };  
});  
  
app.directive('username', function($q, $timeout) {  
 return {  
 require: 'ngModel',  
 link: function(scope, elm, attrs, ctrl) {  
 var usernames = ['Jim', 'John', 'Jill', 'Jackie'];  
  
 ctrl.$asyncValidators.username = function(modelValue, viewValue) {  
  
 if (ctrl.$isEmpty(modelValue)) {  
 // consider empty model valid  
 return $q.when();  
 }  
  
 var def = $q.defer();  
  
 $timeout(function() {  
 // Mock a delayed response  
 if (usernames.indexOf(modelValue) === -1) {  
 // The username is available  
 def.resolve();  
 } else {  
 def.reject();  
 }  
  
 }, 2000);  
  
 return def.promise;  
 };  
 }  
 };  
});