**Angular WorkShop(Introduction)**

AngularJS is a JavaScript Framework

AngularJS is a JavaScript framework written in JavaScript.

AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

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

**Example #01**

**Ng-Model in Angular JS**

The **ng**-**model** Attribute. As discussed in the introduction to this chapter, the **ng**-**model** attribute is used to bind the data in your **model** to the view presented to the user. The **ng**-**model** attribute is used for, Binding controls such as input, text area and selects in the view into the **model**

**ng-app**

The **ng-app** directive tells AngularJS that the <div> element is the "owner" of an AngularJS **application**.

**Example No #01**

<div ng-app="" ng-init="firstName='John'">  
  
<p>The name is <span ng-bind="firstName"></span></p>  
  
</div>

Example No#02

<!DOCTYPE html>  
<html lang="en-US">  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>Name : <input type="text" ng-model="name"></p>  
  <h1>Hello {{name}}</h1>  
</div>  
  
</body>  
</html>

**data-ng-app**

<div data-ng-app="" data-ng-init="firstName='John'">  
  
<p>The name is <span data-ng-bind="firstName"></span></p>  
  
</div>

## AngularJS Expressions

AngularJS expressions are written inside double braces: **{{ expression }}**.

AngularJS will "output" data exactly where the expression is written:

<!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>My first expression: {{ 5 + 5 }}</p>  
</div>  
  
</body>  
</html>

**ng-Bind**

The **ng-bind** directive binds the content of the <p> element to the application variable **name**.

<!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>Name: <input type="text" ng-model="name"></p>  
  <p>{{name}}</p>  
</div>  
  
</body>  
</html>

## The ng-disabled Directive

The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements.

he **ng-disabled** directive binds the application data **mySwitch** to the HTML button's **disabled** attribute.

The **ng-model** directive binds the value of the HTML checkbox element to the value of **mySwitch**.

If the value of **mySwitch** evaluates to **true**, the button will be disabled:

<div ng-app="" ng-init="mySwitch=true">  
  
<p>  
<button ng-disabled="mySwitch">Click Me!</button>  
</p>  
  
<p>  
<input type="checkbox" ng-model="mySwitch">Button  
</p>  
  
<p>  
{{ mySwitch }}  
</p>  
  
</div>

## The ng-show Directive

**The ng-show directive shows or hides an HTML element.**

### **AngularJS Example**

<div ng-app="">  
  
<p ng-show="true">I am visible.</p>  
  
<p ng-show="false">I am not visible.</p>  
  
</div>

## The ng-hide Directive

The **ng-hide** directive hides or shows an HTML element

<div ng-app="">  
  
<p ng-hide="true">I am not visible.</p>  
  
<p ng-hide="false">I am visible.</p>  
  
</div>

## Mouse Events

Mouse events occur when the cursor moves over an element, in this order:

1. ng-mouseover
2. ng-mouseenter
3. ng-mousemove
4. ng-mouseleave

Or when a mouse button is clicked on an element, in this order:

1. ng-mousedown
2. ng-mouseup
3. ng-click

You can add mouse events on any HTML element.

Example No #01 ng-mousemove

<div ng-app="myApp" ng-controller="myCtrl">  
  
<h1 ng-mousemove="count = count + 1">Mouse over me!</h1>  
  
<h2>{{ count }}</h2>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.count = 0;  
});  
</script>

## The ng-click Directive

The ng-click directive defines AngularJS code that will be executed when the element is being clicked.

<div ng-app="myApp" ng-controller="myCtrl">  
  
<button ng-click="count = count + 1">Click me!</button>  
  
<p>{{ count }}</p>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.count = 0;  
});  
</script>

Example

div ng-app="myApp" ng-controller="myCtrl">  
  
<button ng-click="myFunction()">Click me!</button>  
  
<p>{{ count }}</p>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.count = 0;  
  $scope.myFunction = function() {  
    $scope.count++;  
  }  
});  
</script>

**Show Me Functions**

The showMe variable starts out as the Boolean value false.

The myFunc function sets the showMe variable to the opposite of what it is, by using the ! (not) operator.

<div ng-app="myApp" ng-controller="myCtrl">  
  
<button ng-click="myFunc()">Click Me!</button>  
  
<div ng-show="showMe">  
  <h1>Menu:</h1>  
  <div>Pizza</div>  
  <div>Pasta</div>  
  <div>Pesce</div>  
</div>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.showMe = false;  
  $scope.myFunc = function() {  
    $scope.showMe = !$scope.showMe;  
  }  
});  
</script>

## $event Object

You can pass the $event object as an argument when calling the function.

The $event object contains the browser's event object:

div ng-app="myApp" ng-controller="myCtrl">  
  
<h1 ng-mousemove="myFunc($event)">Mouse Over Me!</h1>  
  
<p>Coordinates: {{x + ', ' + y}}</p>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
  $scope.myFunc = function(myE) {  
    $scope.x = myE.clientX;  
    $scope.y = myE.clientY;  
  }  
});  
</script>

# **AngularJS HTML DOM**

## The ng-disabled Directive

The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements.

<div ng-app="" ng-init="mySwitch=true">  
  
<p>  
<button ng-disabled="mySwitch">Click Me!</button>  
</p>  
  
<p>  
<input type="checkbox" ng-model="mySwitch">Button  
</p>  
  
<p>  
{{ mySwitch }}  
</p>  
  
</div>

<p>  
<button disabled>Click Me!</button>  
</p>

<p>  
<button>Click Me!</button>  
</p>

## The ng-show Directive

The **ng-show** directive shows or hides an HTML element.

<div ng-app="">  
  
<p ng-show="true">I am visible.</p>  
  
<p ng-show="false">I am not visible.</p>  
  
</div>

<div ng-app="" ng-init="hour=13">  
  
<p ng-show="hour > 12">I am visible.</p>  
  
</div>

## The ng-hide Directive

The **ng-hide** directive hides or shows an HTML element.

### **AngularJS Example**

<div ng-app="">  
  
<p ng-hide="true">I am not visible.</p>  
  
<p ng-hide="false">I am visible.</p>  
  
</div>

# **AngularJS Form Validation**

## Form Validation

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

## 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>

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

<style>

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

</style>

<style>

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

</style>

## 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.

<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>

## Validation Example

<!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/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>