|  |  |  |  |
| --- | --- | --- | --- |
| **Introduction to Angular JS**  Basics and Syntax of Angular JS, Features, Advantages, Application Structure, Basics of routes and navigation, MVC with Angular JS, Services |  | |  |
| **Angular JS in Details**  Modules, Directives, Routes, Angular JS Forms and Validations, Data binding, Creating single page website using Angular JS | |  | |

Adv :

### Two-way data binding

### DOM manipulation

### Improved server performance

### Responsive web

### Plain HTML templates

DisAdv:

### JavaScript support mandatory

### scopes

### MVC

# AngularJS Introduction

AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <script> tag.

AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

<p>Name: <input type="text" ng-model="name"></p>

AngularJS extends HTML with **ng-directives**.

The **ng-app** directive defines an AngularJS application.

The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

The **ng-bind** directive binds application data to the HTML view.

<!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 ng-bind="name"></p>  
</div>  
  
</body>  
</html>

## AngularJS Expressions/Data Binding

<p>My first expression: {{ 5 + 5 }}</p>

<p>{{name}}</p>

<input style="background-color:{{myCol}}" ng-model="myCol">

# AngularJS Modules

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Controllers always belong to a module.

<div ng-app="**myApp**" ng-controller=**"myCtrl"**>  
{{ firstName + " " + lastName }}  
</div>  
  
<script>  
  
var app = angular.module(**"myApp"**, []);  
  
app.controller(**"myCtrl"**, function($scope) {  
  $scope.firstName = "John";  
  $scope.lastName = "Doe";  
});  
  
</script>

## Repeating HTML Elements

<div ng-app="" ng-init="names=['Jani','Hege','Kai']">  
  <ul>  
    <li ng-repeat="x in names">  
      {{ x }}  
    </li>  
  </ul>  
</div>

## AngularJS Directives

AngularJS directives are extended HTML attributes with the prefix ng-.

The ng-app directive initializes an AngularJS application.

The ng-init directive initializes application data.

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

<div ng-app="" ng-init="firstName='John'">  
  
<p>Name: <input type="text" ng-model="firstName"></p>  
<p>You wrote: {{ firstName }}</p>  
  
</div>

## What is Routing in AngularJS?

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

The ngRoute module routes your application to different pages without reloading the entire application.

<body ng-app="myApp">  
  
<p><a href="#/!">Main</a></p>  
  
<a href="#!red">Red</a>  
<a href="#!green">Green</a>  
<a href="#!blue">Blue</a>  
  
<div ng-view></div>  
  
<script>  
var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
  $routeProvider  
  .when("/", {  
    templateUrl : "main.htm"  
  })  
  .when("/red", {  
    templateUrl : "red.htm"  
  })  
  .when("/green", {  
    templateUrl : "green.htm"  
  })  
  .when("/blue", {  
    templateUrl : "blue.htm"  
  });  
});  
</script>  
</body>

## Checkbox

A checkbox has the value true or false. Apply the ng-model directive to a checkbox, and use its value in your application.

<div ng-app="">

<form>

ABC<input type="checkbox" ng-model="myVar">

<br>

XYZ<input type="checkbox" ng-model="myVar1">

</form>

<h1 ng-show="myVar">ABC</h1>

<h1 ng-show="myVar1">XYZ</h1>

</div>

## Radiobuttons

Bind radio buttons to your application with the ng-model directive.

Radio buttons with the same ng-model can have different values, but only the selected one will be used.

<form>  
  Pick a topic:  
  <input type="radio" ng-model="myVar" value="dogs">Dogs  
  <input type="radio" ng-model="myVar" value="tuts">Tutorials  
  <input type="radio" ng-model="myVar" value="cars">Cars  
</form>

<div ng-switch="myVar">

<div ng-switch-when="dogs">

<h1>Dogs</h1>

<p>Welcome to a world of dogs.</p>

</div>

<div ng-switch-when="tuts">

<h1>Tutorials</h1>

<p>Learn from examples.</p>

</div>

<div ng-switch-when="cars">

<h1>Cars</h1>

<p>Read about cars.</p>

</div>

</div>

## Selectbox

Bind select boxes to your application with the ng-model directive.

The property defined in the ng-model attribute will have the value of the selected option in the selectbox.

<form>

Select a topic:

<select ng-model="myVar">

<option value="--Select--">

<option value="dogs">Dogs

<option value="tuts">Tutorials

<option value="cars">Cars

</select>

</form>

<div ng-switch="myVar">

<div ng-switch-when="dogs">

<h1>Dogs</h1>

<p>Welcome to a world of dogs.</p>

</div>

<div ng-switch-when="tuts">

<h1>Tutorials</h1>

<p>Learn from examples.</p>

</div>

<div ng-switch-when="cars">

<h1>Cars</h1>

<p>Read about cars.</p>

</div>

</div>

## An AngularJS Form Example

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

<form novalidate>

First Name:<br>

<input type="text" ng-model="user.firstName"><br>

Last Name:<br>

<input type="text" ng-model="user.lastName">

<br><br>

<button ng-click="reset()">RESET</button>

</form>

<p>{{user.firstName}}</p>

<p>form = {{user}}</p>

<p>master = {{master}}</p>

</div>

<script>

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

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

$scope.master = {firstName:"a", lastName:"b"};

$scope.reset = function() {

$scope.user = angular.copy($scope.master);

};

$scope.reset();

});

</script>

## Required

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

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

**Email :**

<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>

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
* $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:

* $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.

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

## Custom Validation

<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>

### 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.