AngularJS 简介

AngularJS 诞生于 2009 年，由 Misko Hevery 等人创建，后为 Google 所收购。是一款优秀的前端 JS 框架，已经被用于 Google 的多款产品当中。AngularJS 有着诸多特性，最为核心的是：MVVM（Model-View-ViewModel）、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。

AngularJS 是一个 JavaScript 框架。它是一个以 JavaScript 编写的库。

AngularJS 是以一个 JavaScript 文件形式发布的，可通过 script 标签添加到网页中：<script src="libs/angular.min.js"></script>

AngularJS 通过指令扩展了 HTML，且通过表达式绑定数据到 HTML。

AngularJS 能做什么

AngularJS 是专门为应用程序设计的 HTML。
AngularJS 使得开发现代的单一页面应用程序变得更加容易。

1）AngularJS 把应用程序数据绑定到 HTML 元素。
2）AngularJS 可以克隆和重复 HTML 元素。
3）AngularJS 可以隐藏和显示 HTML 元素。
4）AngularJS 可以在 HTML 元素"背后"添加代码。
5）AngularJS 支持输入验证。

AngularJS 数据绑定

AngularJS 通过 ng-directives(指令) 扩展了 HTML。
ng-app 指令定义一个 AngularJS 应用程序。
ng-model 指令把元素值(比如输入域的值)绑定到应用程序。
ng-bind 指令把应用程序数据绑定到 HTML 视图。

```
// 声明这个 div 是一个 angular 应用程序
<div ng-app="">
  <p> 在输入框中尝试输入: </p>
  // 将文本框内输入的的值绑定到应用程序
  <p> 姓名: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
```

AngularJS 初始化绑定数据

ng-init 指令用来初始化 AngularJS 应用程序变量。
多个需要初始化的数据之间用分号 (;) 隔开。

```
<div ng-app="" ng-init="firstName='John';secondName='Jane'">
    <p>姓名为 <span ng-bind="firstName"></span></p>
    <p>姓名为 <span ng-bind="secondName"></span></p>
</div>
```

也可以用控制器来初始化数据。

AngularJS 扩展属性

HTML5 允许扩展的（自制的）属性，以 data- 开头。

AngularJS 属性以 ng- 开头，但是您可以使用 data-ng- 来让网页对 HTML5 有效。

```
<div data-ng-app="" data-ng-init="firstName='John'">
    <p>姓名为 <span data-ng-bind="firstName"></span></p>
</div>
```

AngularJS 表达式

AngularJS 表达式写在双大括号内：{{ expression }}。
AngularJS 将在表达式书写的位置"输出"数据。
AngularJS 表达式可以包含文字、运算符和变量。

```
<div ng-app="" ng-init="firstValue=3;secondValue=7">
  <p>我的第一个表达式：  {{ 5 + 5 }}</p>
  <p>我的第二个表达式：{{firstValue * secondValue}}</p>
   拼接字符串 <span ng-bind="firstValue+' '+secondValue"></span>
</div>
```

AngularJS 对象

AngularJS 也可以使用对象。

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">
  <p>姓为 {{ person.firstName }}</p>
  <p>名为 <span ng-bind="person.lastName"></span></p>
</div>
```

AngularJS 也可以使用数组。

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">
   <p>第三个值为 {{ points[2] }}</p>
   <p>第四个值为 <span ng-bind="points[4]"></span></p>
</div>
```

AngularJS 重复元素

AngularJS的 ng-repeat 指令会重复一个 HTML 元素:

```
<div ng-app="" ng-init="names=[
  {name:'Jani',country:'Norway'},
  {name:'Hege',country:'Sweden'},
  {name:'Kai',country:'Denmark'}]">
  <p>循环对象:  </p>
  <ul>
    <li ng-repeat="x in names">
      {{ ($index + 1) + ', ' + x.name + ', ' + x.country }}
    </li>
  </ul>
</div>
```

AngularJS 控制器

AngularJS 模块（Module） 定义了 AngularJS 应用。
AngularJS 控制器（Controller） 用于控制 AngularJS 应用。
ng-app 指令定义了应用，ng-controller 定义了控制器。

```
<div ng-app="myApp" ng-controller="myCtrl">
   名：<input type="text" ng-model="firstName"><br>
   姓：<input type="text" ng-model="lastName"><br>
   姓名：{{fullName()}}
</div>

<script>
  var app = angular.module('myApp', []);// 数组通常包含从属模块
  app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";// 数据初始化
    $scope.lastName= "Doe";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
     }
  });
</script>
```

AngularJS 控制器

应用解析:

1）AngularJS 应用程序由 ng-app 定义。应用程序在 内运行（范围）。
2）ng-controller="myCtrl" 是一个 AngularJS 指令。用于定义一个控制器。
（一个模块 ng-app 下的多个控制器都操作了同一个 $scope 的一个对象，以后面的为准，即覆盖了前面的）
3）myCtrl 函数是一个 JavaScript 函数。
4）AngularJS 使用 $scope 对象来调用控制器。
5）在 AngularJS 中，$scope 是一个应用对象（属于应用变量和函数）。
6）控制器的 $scope （相当于作用域、控制范围）用来保存 AngularJS Model(模型）的对象。
7）控制器在作用域中创建了两个属性 (firstName 和 lastName)。
8）ng-model 指令绑定输入域到控制器的属性（firstName 和 lastName）。
9）控制器也可以有方法（变量和函数）:

AngularJS 过滤器

AngularJS 过滤器可用于转换数据:
过滤器可以通过一个管道字符（|）和一个过滤器添加到表达式中。

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

AngularJS 过滤器

```
<p>过滤姓为大写 {{ lastName | uppercase }}</p>
<p>过滤数字为货币格式 = {{ price | currency }}</p>
将对象按照某一个属性排序:
   <li ng-repeat="x in names | orderBy:'country'">
      {{ x.name + ', ' + x.country }}
   </li>
将对象中包含 test 的对象搜索出来:
   <li ng-repeat="x in names | filter:test | orderBy:'country'">
      {{ (x.name | uppercase) + ', ' + x.country }}
   </li>
```

AngularJS $HTTP

AngularJS $http 是一个用于读取 web 服务器上数据的服务（类似于 ajax）。
$http.get(url) 是用于读取服务器数据的函数。
下面例子是请求服务器的数据，然后显示到列表中。

```
<li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
<script>
   var app = angular.module('myApp', []);
   app.controller('customersCtrl', function($scope, $http){
   $http.get("http://www.runoob.com/try/angularjs/data/Customers_JSON.
php")
    .success(function(response) {$scope.names = response.records;});
});
</script>
```

AngularJS HTML DOM

AngularJS 为 HTML DOM 元素的属性提供了绑定应用数据的指令。
DOM 元素可以直接使用这些绑定的指令来改变样式。

```
<div ng-init="mySwitch=true">
<p><button ng-disabled="mySwitch" ng-click="mySwitch = ! mySwitch">点我 !</button></p>
<p><input type="checkbox" ng-model="mySwitch"/>按钮</p>
<p>{{ mySwitch }}</p>
<p ng-show="mySwitch">我是可见的。 </p>
</div>
```

AngularJS 表单验证

前面已经介绍了数据绑定和控制器，我们也可以通过这两种方法来实现表单的恢复默认值和数据验证。

```
<form ng-app="myApp" ng-controller="validateCtrl"
name="myForm" novalidate>
<p>邮箱：<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">邮箱是必须的。 </span>
<span ng-show="myForm.email.$error.email">非法的邮箱地址。 </span>
</span>
</p>
</form>
<script>
var app = angular.module('myApp', []);
app.controller('validateCtrl', function($scope) {
    $scope.email = 'john.doe@gmail.com';
});
</script>
```

AngularJS 表单验证

HTML 表单属性 novalidate 用于禁用浏览器默认的验证。novalidate 属性在应用中不是必须的，但是你需要在 AngularJS 表单中使用，用于重写标准的 HTML5 验证。

属性	描述
$dirty	表单有填写记录
$valid	字段内容合法的
$invalid	字段内容是非法的
$pristine	表单没有填写记录

AngularJS API

AngularJS 全局 API 用于执行常见任务的 JavaScript 函数集合
以下列出了一些通用的 API 函数:

API	描述
angular.lowercase()	转换字符串为小写
angular.uppercase()	转换字符串为大写
angular.isString()	判断给定的对象是否为字符串，如果是返回 true。
angular.isNumber()	判断给定的对象是否为数字，如果是返回 true。

这些函数的用法类似: $scope.x2 = angular.lowercase($scope.x1);

AngularJS 包含 HTML

使用 AngularJS, 你可以使用 ng-include 指令来包含 HTML 内容:

```
<div >
  <div ng-include="'myUsers_List.htm'"></div>
  <div ng-include="'myUsers_Form.htm'"></div>
</div>
```

AngularJS Bootstrap

AngularJS 的首选样式表是 Twitter Bootstrap， Twitter Bootstrap 是目前最受欢迎的前端框架。附上该框架的教程地址（http://www.runoob.com/bootstrap/bootstrap-tutorial.html）

如果在你的 AngularJS 应用中要加入 Twitter Bootstrap，你需要在你的 <head> 元素中添加如下代码：

```
<link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css"
>
```

如果站点在国内，建议使用百度静态资源库的 Bootstrap，代码如下：

```
<link rel="stylesheet"
href="//apps.bdimg.com/libs/bootstrap/3.2.0/css/bootstrap.min.css">
```

AngularJS Bootstrap 样例

Bootstrap 类解析: （它包含的不止这么多内容，这些是样例中用到的）

元素	Bootstrap 类	定义
	container	内容容器
<table>	table	表格
<table>	table-striped	带条纹背景的表格
<button>	btn	按钮
<button>	btn-success	成功按钮
	glyphicon	字形图标
	glyphicon-pencil	铅笔图标
	glyphicon-user	用户图标
	glyphicon-save	保存图标
<form>	form-horizontal	水平表格
	form-group	表单组
<label>	control-label	控制器标签
<label>	col-sm-2	跨越 2 列
	col-sm-10	跨越 10 列

AngularJS 视图

AngularJS 支持通过在单个页面上的多个视图的单页应用。（单页应用：指在浏览器中运行的应用，它们在使用期间不会重新加载页面（刷新局部）。在开发单页应用时第一个要处理的问题就是页面结构化，由于多个功能集中在一个页面呈现，就必然需要考虑如何实现多个视图布局？如何实现视图之间动画切换？等问题。视图是单页应用开发中最常见的模块，通常在一个单页应用中，会有多个视图存在，每一个视图都可以处理一部分业务功能，所有视图的功能集就是单页应用所能处理业务的最大能力。常见的几种视图：单视图层，封面型，侧边栏）要做到这一点 AngularJS 提供 ng-view 和 ng-template 指令，以及 $routeProvider 服务。

ng-view 标记只是简单地创建一个占位符，是一个相应的视图（HTML 或 ng-template 视图），可以根据配置来放置。

```
<div ng-app="mainApp">
    <div ng-view></div>
</div>
```

AngularJS 视图

ng-template 指令是用来创建使用 script 标签的 HTML 视图。它包含一个用于由 $routeProvider 映射控制器视图的“id”属性。

```
<div ng-app="mainApp">
   <script type="text/ng-template" id="addStudent.html">
      <h2> Add Student </h2>
         {{message}}
   </script>
</div>
```

$routeProvider 是组网址的配置，将它们映射相应的 HTML 页面或 ng-template，并附加一个控制器使用相同键的服务。

AngularJS 视图

```
var mainApp = angular.module("mainApp", ['ngRoute']);

      mainApp.config(['$routeProvider',
         function($routeProvider) {
            $routeProvider.
               when('/addStudent', {
                  templateUrl: 'addStudent.html',
                  controller: 'AddStudentController'
               }).
               when('/viewStudents', {
                  templateUrl: 'viewStudents.html',
                  controller: 'ViewStudentsController'
               }).
               otherwise({
                  redirectTo: '/addStudent'
               });
         }]);
```

“otherwise” 是用来设置的默认视图;

“conlloer” 是用来设置该视图对应的控制器;

AngularJS 服务

AngularJS 支持使用服务的体系结构“关注点分离”的概念。服务是 JavaScript 函数，并负责只做一个特定的任务。这也使得他们即维护和测试的单独实体。控制器，过滤器可以调用它们作为需求的基础。服务使用 AngularJS 的依赖注入机制注入正常。

AngularJS 提供例如许多内在的服务，如： $http，$route，$window，$location 等。每个服务负责例如一个特定的任务， $http 是用来创建 AJAX 调用，以获得服务器的数据。 $route 用来定义路由信息等。内置的服务总是前缀 $ 符号。

有两种方法来创建服务：工厂，服务。

AngularJS 服务

```
var mainApp = angular.module("mainApp", []);
```

使用工厂方法，我们先定义一个工厂，然后分配方法给它。

```
      mainApp.factory('MathService', function() {
         var factory = {};
         factory.multiply = function(a, b) {
            return a * b
         }
         return factory;
      });
```

使用服务的方法，我们定义一个服务，然后分配方法。还注入已经可用的服务。

```
mainApp.service('CalcService', function(MathService){
    this.square = function(a) {
       return MathService.multiply(a,a);
    }
});
```

然后我们就可以在控制器中使用这个服务。 服务 CalcService 中使用了工厂 MathService 。

AngularJS 依赖注入

AngularJS 中的各函数之间的依赖能在需要时被导入。因为它不必在组件中去主动寻找和获取依赖，而是由外界将依赖传入。每一个 AngularJS 应用都有一个注入器 (injector) 用来处理依赖的创建。注入器是一个负责查找和创建依赖的服务定位器。

```
angular.module('myModule', []).factory('greeter', function($window) {
    return {
      greet: function(text) {
        $window.alert(text);
      }
    };
  }).
var injector = angular.injector('myModule');
var greeter = injector.get('greeter');
```

但是同样注入器需要在应用中传递。

AngularJS 依赖注入

AngularJS 最简单的处理依赖的方法，就是假设函数的参数名就是依赖的名字。

```
function MyController($scope, greeter) {}
```

$scope 和 greeter 是需要注入到函数中的依赖。

用了这种方法就不能使用 JavaScript 中一些用来缩短的 JS 的类库了，因为它们会改变变量名。要允许压缩类库重命名函数参数，同时注入器又能正确处理依赖的话，函数需要使用 $inject 属性。这个属性是一个包含依赖的名称的数组。

```
var MyController = function(renamed$scope, renamedGreeter) {}
MyController.$inject = ['$scope', 'greeter'];
```

注意 $inject 标记里的值和函数声明的参数是对应的。

AngularJS 自定义标签

自定义指令中使用 AngularJS 扩展 HTML 的功能。以下列元素的类型来创建自定义指令。

Element directives - 指令遇到时激活一个匹配的元素。

Attribute - - 指令遇到时激活一个匹配的属性。

CSS - - 指令遇到时激活匹配 CSS 样式。

Comment - - 指令遇到时激活匹配的注释。

使用自定义的指令首先创建自定义的 HTML 标签。

```
<student name="Mahesh"></student><br/>
<student name="Piyush"></student>
```

AngularJS 自定义标签

定义自定义指令来处理上面的自定义 HTML 标签。

```
var mainApp = angular.module("mainApp", []);
      mainApp.directive('student', function() {
         var directive = {};
         directive.restrict = 'E';
         directive.template = "Student: <b>{{student.name}}</b> , Roll No:
<b>{{student.rollno}}</b>";
         directive.scope = {
            student : "=name"
         }
         directive.compile = function(element, attributes) {
            element.css("border", "1px solid #cccccc");
            var linkFunction = function($scope, element, attributes) {
               element.html("Student: <b>"+$scope.student.name +"</b> , Roll
No: <b>"+$scope.student.rollno+"</b><br/>");
               element.css("background-color", "#ff00ff");
            }
            return linkFunction;
         }
         return directive;
      });
```