AngularJS 教程

AngularJS 通过新的属性和表达式扩展了 HTML。
AngularJS 可以构建一个单一页面应用程序（SPAs：Single Page Applications）。
AngularJS 学习起来非常简单。
现在开始学习 AngularJS!

每个章节都有相应的实例
在每个章节中，您可以在线编辑实例，然后点击按钮查看结果。

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>
</head>
<body>

<div ng-app="">
    <p>名字 : <input type="text" ng-model="name"></p>
    <h1>Hello {{name}}</h1>
</div>

</body>
</html>
```

尝试一下 »

阅读本教程前，您需要了解的知识：
在开始学习 AngularJS 之前，您需要具备以下基础知识：
HTML
CSS
JavaScript

AngularJS 历史
AngularJS 是比较新的技术，版本 1.0 是在 2012 年发布的。
AngularJS 是由 Google 的员工 Mško Hevery 从 2009 年开始着手开发。
这是一个非常好的构想，该项目目前已由 Google 正式支持，有一个全职的开发团队继续开发和维护这个库。

AngularJS 实例
本教程包含了大量的 AngularJS 实例！
AngularJS 实例

AngularJS 参考手册
参考手册包含了本教程中使用到的所有指令和过滤器。
AngularJS 参考手册

AngularJS 简介

AngularJS 是一个 **JavaScript 框架**。它可通过 <script> 标签添加到 HTML 页面。
AngularJS 通过 **指令** 扩展了 HTML，且通过 **表达式** 绑定数据到 HTML。

AngularJS 是一个 JavaScript 框架
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="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>
```

我们建议把脚本放在 <body> 元素的底部。
这会提高网页加载速度，因为 HTML 加载不受制于脚本加载。

各个 angular.js 版本下载： https://github.com/angular/angular.js/releases

AngularJS 扩展了 HTML
AngularJS 通过 **ng-directives** 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。

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>
</head>
```

```
<body>

<div ng-app="">
     <p>名字 ： <input type="text" ng-model="name"></p>
     <h1>Hello {{name}}</h1>
</div>

</body>
</html>
```

尝试一下 »

实例讲解：
当网页加载完毕，AngularJS 自动开启。
ng-app 指令告诉 AngularJS， 元素是 AngularJS **应用程序** 的"所有者"。
ng-model 指令把输入域的值绑定到应用程序变量 **name**。
ng-bind 指令把应用程序变量 name 绑定到某个段落的 innerHTML。

如果您移除了 **ng-app** 指令，HTML 将直接把表达式显示出来，不会去计算表达式的结果。

什么是 AngularJS?
AngularJS 使得开发现代的单一页面应用程序（SPAs：Single Page Applications）变得更加容易。
AngularJS 把应用程序数据绑定到 HTML 元素。
AngularJS 可以克隆和重复 HTML 元素。
AngularJS 可以隐藏和显示 HTML 元素。
AngularJS 可以在 HTML 元素"背后"添加代码。
AngularJS 支持输入验证。

AngularJS 指令
正如您所看到的，AngularJS 指令是以 **ng** 作为前缀的 HTML 属性。
ng-init 指令初始化 AngularJS 应用程序变量。

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John'">

<p>姓名为 <span ng-bind="firstName"></span></p>

</div>
```

尝试一下 »

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。

带有有效的 HTML5：

AngularJS 实例

```
<div data-ng-app="" data-ng-init="firstName='John'">

<p>姓名为 <span data-ng-bind="firstName"></span></p>

</div>
```

尝试一下 »

AngularJS 表达式
AngularJS 表达式写在双大括号内：**{{ expression }}**。
AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。
AngularJS 将在表达式书写的位置"输出"数据。
AngularJS 表达式 很像 **JavaScript 表达式**：它们可以包含文字、运算符和变量。
实例 {{ 5 + 5 }} 或 {{ firstName + " " + lastName }}

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>
</head>
<body>

<div ng-app="">
     <p>我的第一个表达式： {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

尝试一下 »

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

AngularJS 实例

```
<div ng-app="myApp" ng-controller="myCtrl">

名: <input type="text" ng-model="firstName"><br>
姓: <input type="text" ng-model="lastName"><br>
<br>
姓名: {{firstName + " " + lastName}}

</div>

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

尝试一下 »

AngularJS 模块定义应用:

AngularJS 模块

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

AngularJS 控制器控制应用:

AngularJS 控制器

```
app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";
    $scope.lastName= "Doe";
});
```

在接下来的教程中你将学习到更多的应用和模块的知识。

AngularJS 表达式

AngularJS 使用 **表达式** 把数据绑定到 HTML。

AngularJS 表达式
AngularJS 表达式写在双大括号内：**{{ expression }}**。
AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。
AngularJS 将在表达式书写的位置"输出"数据。
AngularJS 表达式 很像 **JavaScript 表达式**：它们可以包含文字、运算符和变量。
实例 {{ 5 + 5 }} 或 {{ firstName + " " + lastName }}

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>
</head>
<body>

<div ng-app="">
     <p>我的第一个表达式: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

尝试一下 »

AngularJS 数字
AngularJS 数字就像 JavaScript 数字：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： {{ quantity * cost }}</p>

</div>
```

尝试一下 »

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： <span ng-bind="quantity * cost"></span></p>

</div>
```

尝试一下 »

使用 **ng-init** 不是很常见。您将在控制器一章中学习到一个更好的初始化数据的方式。

AngularJS 字符串

AngularJS 字符串就像 JavaScript 字符串：

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： {{ firstName + " " + lastName }}</p>

</div>
```

尝试一下 »

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： <span ng-bind="firstName + ' ' + lastName"></span></p>

</div>
```

尝试一下 »

AngularJS 对象

AngularJS 对象就像 JavaScript 对象：

AngularJS 实例

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 {{ person.lastName }}</p>

</div>
```

尝试一下 »

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 <span ng-bind="person.lastName"></span></p>

</div>
```

尝试一下 »

AngularJS 数组

AngularJS 数组就像 JavaScript 数组：

AngularJS 实例

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 {{ points[2] }}</p>

</div>
```

尝试一下 »

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 <span ng-bind="points[2]"></span></p>
```

```
</div>
```

尝试一下 »

AngularJS 表达式 与 JavaScript 表达式
类似于 JavaScript 表达式，AngularJS 表达式可以包含字母，操作符，变量。
与 JavaScript 表达式不同，AngularJS 表达式可以写在 HTML 中。
与 JavaScript 表达式不同，AngularJS 表达式不支持条件判断，循环及异常。
与 JavaScript 表达式不同，AngularJS 表达式支持过滤器。

AngularJS 指令

AngularJS 通过被称为 **指令** 的新属性来扩展 HTML。
AngularJS 通过内置的指令来为应用添加功能。
AngularJS 允许你自定义指令。

AngularJS 指令
AngularJS 指令是扩展的 HTML 属性，带有前缀 **ng-**。
ng-app 指令初始化一个 AngularJS 应用程序。
ng-init 指令初始化应用程序数据。
ng-model 指令把元素值（比如输入域的值）绑定到应用程序。
完整的指令内容可以参阅 AngularJS 参考手册。

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John'">

    <p>在输入框中尝试输入：</p>
    <p>姓名：<input type="text" ng-model="firstName"></p>
    <p>你输入的为： {{ firstName }}</p>

</div>
```

尝试一下 »

ng-app 指令告诉 AngularJS， 元素是 AngularJS **应用程序** 的"所有者"。
数据绑定
上面实例中的 **{{ firstName }}** 表达式是一个 AngularJS 数据绑定表达式。
AngularJS 中的数据绑定，同步了 AngularJS 表达式与 AngularJS 数据。
{{ firstName }} 是通过 **ng-model="firstName"** 进行同步。
在下一个实例中，两个文本域是通过两个 ng-model 指令同步的：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;price=5">

<h2>价格计算器</h2>

数量： <input type="number"    ng-model="quantity">
价格： <input type="number" ng-model="price">

<p><b>总价：</b> {{ quantity * price }}</p>

</div>
```

尝试一下 »

使用 **ng-init** 不是很常见。您将在控制器一章中学习到一个更好的初始化数据的方式。

重复 HTML 元素
ng-repeat 指令会重复一个 HTML 元素：

AngularJS 实例

```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
  <p>使用 ng-repeat 来循环数组</p>
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
</div>
```

尝试一下 »

ng-repeat 指令用在一个对象数组上：

AngularJS 实例

```
<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">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>
```

尝试一下 »

AngularJS 完美支持数据库的 CRUD（增加Create、读取Read、更新Update、删除Delete）应用程序。
把实例中的对象想象成数据库中的记录。

ng-app 指令
ng-app 指令定义了 AngularJS 应用程序的 **根元素**。
ng-app 指令在网页加载完毕时会**自动引导**（自动初始化）应用程序。
稍后您将学习到 **ng-app** 如何通过一个值（比如 ng-app="myModule"）连接到代码模块。

ng-init 指令
ng-init 指令为 AngularJS 应用程序定义了 **初始值**。
通常情况下，不使用 ng-init。您将使用一个控制器或模块来代替它。
稍后您将学习更多有关控制器和模块的知识。

ng-model 指令
ng-model 指令 **绑定 HTML 元素** 到应用程序数据。
ng-model 指令也可以：
为应用程序数据提供类型验证（number、email、required）。
为应用程序数据提供状态（invalid、dirty、touched、error）。
为 HTML 元素提供 CSS 类。
绑定 HTML 元素到 HTML 表单。

ng-repeat 指令
ng-repeat 指令对于集合中（数组中）的每个项会 **克隆一次 HTML 元素**。

创建自定义的指令
除了 AngularJS 内置的指令外，我们还可以创建自定义指令。
你可以使用 **.directive** 函数来添加自定义的指令。
要调用自定义指令，HTML 元素上需要添加自定义指令名。
使用驼峰法来命名一个指令， **runoobDirective**, 但在使用它时需要以 **-** 分割, **runoob-directive**:

AngularJS 实例

```
<body ng-app="myApp">

<runoob-directive></runoob-directive>

<script>
var app = angular.module("myApp", []);
app.directive("runoobDirective",   function() {
    return {
          template : "<h1>自定义指令!</h1>"
    };
  });
</script>

</body>
```

尝试一下 »

你可以通过以下方式来调用指令：
元素名
属性
类名
注释
以下实例方式也能输出同样结果:

元素名

```
<runoob-directive></runoob-directive>
```

尝试一下 »

属性

```
<div runoob-directive></div>
```

尝试一下 »

类名

```
<div class="runoob-directive"></div>
```

尝试一下 »

注释

```
<!-- directive: runoob-directive -->
```

尝试一下 »

限制使用

你可以限制你的指令只能通过特定的方式来调用。

实例

通过添加 **restrict** 属性,并设置值为 "A", 来设置指令只能通过属性的方式来调用:

```
var app = angular.module("myApp", []);
app.directive("runoobDirective",   function() {
    return {
          restrict : "A",
          template : "<h1>自定义指令!</h1>"
    };
  });
```

尝试一下 »

restrict 值可以是以下几种:

E 作为元素名使用

A 作为属性使用

C 作为类名使用

M 作为注释使用

restrict 默认值为 EA, 即可以通过元素名和属性名来调用指令。

AngularJS 控制器

AngularJS 控制器 **控制** AngularJS 应用程序的数据。

AngularJS 控制器是常规的 **JavaScript 对象**。

AngularJS 控制器

AngularJS 应用程序被控制器控制。

ng-controller 指令定义了应用程序控制器。

控制器是 **JavaScript 对象**，由标准的 JavaScript **对象的构造函数** 创建。

AngularJS 实例

```
<div ng-app="myApp" ng-controller="myCtrl">

名: <input type="text" ng-model="firstName"><br>
  姓: <input type="text" ng-model="lastName"><br>
<br>
  姓名: {{firstName + " " + lastName}}

</div>

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

尝试一下 »

应用解析:

AngularJS 应用程序由 **ng-app** 定义。应用程序在 内运行。

ng-controller="myCtrl" 属性是一个 AngularJS 指令。用于定义一个控制器。

myCtrl 函数是一个 JavaScript 函数。

AngularJS 使用**$scope** 对象来调用控制器。

在 AngularJS 中， $scope 是一个应用对象(属于应用变量和函数)。

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。

控制器在作用域中创建了两个属性 (**firstName** 和 **lastName**)。

ng-model 指令绑定输入域到控制器的属性（firstName 和 lastName）。

控制器方法

上面的实例演示了一个带有 lastName 和 firstName 这两个属性的控制器对象。

控制器也可以有方法（变量和函数）：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="personCtrl">

  名: <input type="text" ng-model="firstName"><br>
  姓: <input type="text" ng-model="lastName"><br>
<br>
  姓名: {{fullName()}}
```

```
</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl',   function($scope) {
      $scope.firstName = "John";
    $scope.lastName   = "Doe";
    $scope.fullName = function() {
          return $scope.firstName + " " + $scope.lastName;
    }
});
</script>
```

尝试一下 »

外部文件中的控制器

在大型的应用程序中，通常是把控制器存储在外部文件中。

只需要把 <script> 标签中的代码复制到名为 personController.js 的外部文件中即可：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
  Last Name: <input type="text" ng-model="lastName"><br>
<br>
  Full Name: {{firstName + " " + lastName}}

</div>

<script src="personController.js"></script>
```

尝试一下 »

其他实例

以下实例创建一个新的控制器文件:

```
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Hege',country:'Sweden'},
        {name:'Kai',country:'Denmark'}
    ];
});
```

保存文件为 namesController.js:

然后，在应用中使用控制器文件:

AngularJS 实例

```
<div ng-app="myApp" ng-controller="namesCtrl">

<ul>
     <li ng-repeat="x in names">
      {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>

<script src="namesController.js"></script>
```

尝试一下 »

AngularJS 过滤器

过滤器可以使用一个管道字符（|）添加到表达式和指令中。

AngularJS 过滤器

AngularJS 过滤器可用于转换数据：

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。

表达式中添加过滤器

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。.

(下面的两个实例，我们将使用前面章节中提到的 person 控制器)

uppercase 过滤器将字符串格式化为大写：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="personCtrl">

<p>姓名为 {{ lastName | uppercase }}</p>

</div>
```

尝试一下 »

lowercase 过滤器将字符串格式化为小写：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="personCtrl">

<p>姓名为 {{ lastName | lowercase }}</p>

</div>
```

尝试一下 »

currency 过滤器

currency 过滤器将数字格式化为货币格式：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="costCtrl">

<input type="number" ng-model="quantity">
<input type="number" ng-model="price">

<p>总价 = {{ (quantity * price) | currency }}</p>

</div>
```

尝试一下 »

向指令添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中。

orderBy 过滤器根据表达式排列数组：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="namesCtrl">

<ul>
    <li ng-repeat="x in names | orderBy:'country'">
      {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>
```

尝试一下 »

过滤输入

输入过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中，该过滤器后跟一个冒号和一个模型名称。

filter 过滤器从数组中选择一个子集：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="namesCtrl">

<p><input type="text" ng-model="test"></p>

<ul>
  <li ng-repeat="x in names |   filter:test | orderBy:'country'">
    {{ (x.name | uppercase) + ', ' + x.country }}
  </li>
</ul>

</div>
```

尝试一下 »

自定义过滤器

以下实例自定义一个过滤器 **reverse**，将字符串反转：

AngularJS 实例

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.msg = "Runoob";
});
app.filter('reverse', function() { //可以注入依赖
    return function(text) {
        return text.split("").reverse().join("");
    }
});
```

尝试一下 »

AngularJS 服务(Service)

AngularJS 中你可以创建自己的服务，或使用内建服务。

什么是服务？
在 AngularJS 中，服务是一个函数或对象，可在你的 AngularJS 应用中使用。
AngularJS 内建了30 多个服务。
有个 **$location** 服务，它可以返回当前页面的 URL 地址。

实例

```
var app = angular.module('myApp', []);
app.controller('customersCtrl',   function($scope, $location) {
    $scope.myUrl = $location.absUrl();
  });
```

尝试一下 »

注意 **$location** 服务是作为一个参数传递到 controller 中。如果要使用它，需要在 controller 中定义。

为什么使用服务？
在很多服务中，比如 $location 服务，它可以使用 DOM中存在的对象，类似 window.location 对象，但 window.location 对象在 AngularJS 应用中有一定的局限性。
AngularJS 会一直监控应用，处理事件变化， AngularJS 使用 **$location** 服务比使用 **window.location** 对象更好。
$location vs window.location

	window.location	$location.service
目的	允许对当前浏览器位置进行读写操作	允许对当前浏览器位置进行读写操作
API	暴露一个能被读写的对象	暴露jquery风格的读写器
是否在AngularJS应用生命周期中和应用整合	否	可获取到应用生命周期内的每一个阶段，并且和$watch整合
是否和HTML5 API的无缝整合	否	是（对低级浏览器优雅降级）
和应用的上下文是否相关	否，window.location.path返回"/docroot/actual/path"	是，$location.path()返回"/actual/path"

$http 服务
$http 是 AngularJS 应用中最常用的服务。 服务向服务器发送请求，应用响应服务器传送过来的数据。

实例
使用 **$http** 服务向服务器请求数据:

```
var app = angular.module('myApp', []);
app.controller('myCtrl',   function($scope, $http) {
    $http.get("welcome.htm").then(function   (response) {
        $scope.myWelcome   = response.data;
    });
});
```

尝试一下 »

以上是一个非常简单的 **$http** 服务实例，更多 **$http** 服务应用请查看 AngularJS Http 教程。

$timeout 服务
AngularJS **$timeout** 服务对应了 JS **window.setTimeout** 函数。

实例
两秒后显示信息:

```
var app = angular.module('myApp', []);
app.controller('myCtrl',   function($scope, $timeout) {
    $scope.myHeader = "Hello   World!";
    $timeout(function () {
          $scope.myHeader = "How are you today?";
    }, 2000);
  });
```

尝试一下 »

$interval 服务

AngularJS **$interval** 服务对应了 JS **window.setInterval** 函数。

实例
每一秒显示信息:

```
var app = angular.module('myApp', []);
app.controller('myCtrl',   function($scope, $interval) {
    $scope.theTime = new   Date().toLocaleTimeString();
    $interval(function () {
          $scope.theTime = new Date().toLocaleTimeString();
    },   1000);
});
```

尝试一下 »

创建自定义服务
你可以创建自定义服务，链接到你的模块中：

创建名为**hexafy** 的服务:

```
  app.service('hexafy', function() {
    this.myFunc = function (x) {
          return x.toString(16);
    }
});
```

要使用自定义服务，需要在定义控制器的时候独立添加，设置依赖关系:

实例
使用自定义的的服务 **hexafy** 将一个数字转换为16进制数:

```
  app.controller('myCtrl', function($scope, hexafy) {
    $scope.hex   = hexafy.myFunc(255);
});
```

尝试一下 »

过滤器中，使用自定义服务
当你创建了自定义服务，并连接到你的应用上后，你可以在控制器，指令，过滤器或其他服务中使用它。

在过滤器 **myFormat** 中使用服务 **hexafy**:

```
  app.filter('myFormat',['hexafy', function(hexafy) {
     return function(x) {
       return   hexafy.myFunc(x);
    };
}]);
```

尝试一下 »

在对象数组中获取值时你可以使用过滤器：

创建服务 **hexafy**:

```
<ul>
<li ng-repeat="x in counts">{{x | myFormat}}</li>
</ul>
```

尝试一下 »

AngularJS XMLHttpRequest

$http 是 AngularJS 中的一个核心服务，用于读取远程服务器的数据。
使用格式：

```
// 简单的 GET 请求，可以改为 POST
$http({
    method: 'GET',
    url: '/someUrl'
}).then(function successCallback(response) {
        // 请求成功执行代码
    }, function errorCallback(response) {
        // 请求失败执行代码
});
```

简写方法
POST 与 GET 简写方法格式：

```
$http.get('/someUrl', config).then(successCallback, errorCallback);
$http.post('/someUrl', data, config).then(successCallback, errorCallback);
```

此外还有以下简写方法：

$http.get
$http.head
$http.post
$http.put
$http.delete
$http.jsonp
$http.patch

更详细内容可参见：https://docs.angularjs.org/api/ng/service/$http

读取 JSON 文件

以下是存储在web服务器上的 JSON 文件：

http://www.runoob.com/try/angularjs/data/sites.php

```
{
    "sites": [
        {
            "Name": "菜鸟教程",
            "Url": "www.runoob.com",
            "Country": "CN"
        },
        {
            "Name": "Google",
            "Url": "www.google.com",
            "Country": "USA"
        },
        {
            "Name": "Facebook",
            "Url": "www.facebook.com",
            "Country": "USA"
        },
        {
            "Name": "微博",
            "Url": "www.weibo.com",
            "Country": "CN"
        }
    ]
}
```

AngularJS $http

AngularJS $http 是一个用于读取web服务器上数据的服务。

$http.get(url) 是用于读取服务器数据的函数。

> **废弃声明 (v1.5)**
>
> v1.5 中`$http` 的 `success` 和 `error` 方法已废弃。使用 `then` 方法替代。

通用方法实例

AngularJS1.5 以上版本 - 实例

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

app.controller('siteCtrl', function($scope, $http) {
    $http({
        method: 'GET',
        url: 'https://www.runoob.com/try/angularjs/data/sites.php'
    }).then(function successCallback(response) {
            $scope.names = response.data.sites;
        }, function errorCallback(response) {
            // 请求失败执行代码
    });

});
```

尝试一下 »

简写方法实例

AngularJS1.5 以上版本 - 实例

```
<div ng-app="myApp" ng-controller="siteCtrl">

<ul>
  <li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('siteCtrl', function($scope, $http) {
  $http.get("http://www.runoob.com/try/angularjs/data/sites.php")
```

```
  .then(function (response) {$scope.names = response.data.sites;});
});
</script>
```

尝试一下 »

AngularJS1.5 以下版本 - 实例

```
<div ng-app="myApp" ng-controller="siteCtrl">

<ul>
  <li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('siteCtrl', function($scope, $http) {
  $http.get("http://www.runoob.com/try/angularjs/data/sites.php")
  .success(function (response) {$scope.names = response.sites;});
});
</script>
```

尝试一下 »

应用解析：

注意：以上代码的 get 请求是本站的服务器，你不能直接拷贝到你本地运行，会存在跨域问题，解决办法就是将 Customers_JSON.php 的数据拷贝到你自己的服务器上，附：PHP Ajax跨域问题最佳解决方案。

AngularJS 应用通过 **ng-app** 定义。应用在 中执行。

ng-controller 指令设置了 **controller 对象** 名。

函数 **customersController** 是一个标准的 JavaScript **对象构造器**。

控制器对象有一个属性: **$scope.names**。

$http.get() 从web服务器上读取静态 **JSON 数据**。

服务器数据文件为： **http://www.runoob.com/try/angularjs/data/sites.php**。

当从服务端载入 JSON 数据时，**$scope.names** 变为一个数组。

以上代码也可以用于读取数据库数据。

AngularJS Select(选择框)

AngularJS 可以使用数组或对象创建一个下拉列表选项。

使用 ng-options 创建选择框

在 AngularJS 中我们可以使用 **ng-option** 指令来创建一个下拉列表，列表项通过对象和数组循环输出，如下实例:

实例

```
<div ng-app="myApp" ng-controller="myCtrl">

<select ng-init="selectedName = names[0]" ng-model="selectedName" ng-options="x for x in names">
</select>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.names = ["Google", "Runoob", "Taobao"];
});
</script>
```

尝试一下 »

ng-init 设置默认选中值。

ng-options 与 ng-repeat

我们也可以使用**ng-repeat** 指令来创建下拉列表：

实例

```
<select>
<option ng-repeat="x in names">{{x}}</option>
</select>
```

尝试一下 »

ng-repeat 指令是通过数组来循环 HTML 代码来创建下拉列表，但 **ng-options** 指令更适合创建下拉列表，它有以下优势：

使用 **ng-options** 的选项是一个对象， **ng-repeat** 是一个字符串。

应该用哪个更好？

假设我们使用以下对象:

```
$scope.sites = [
    {site : "Google", url : "http://www.google.com"},
    {site : "Runoob", url : "http://www.runoob.com"},
    {site : "Taobao", url : "http://www.taobao.com"}
];
```

ng-repeat 有局限性，选择的值是一个字符串：

实例
使用 **ng-repeat**:

```
<select ng-model="selectedSite">
<option ng-repeat="x in sites" value="{{x.url}}">{{x.site}}</option>
</select>

<h1>你选择的是：{{selectedSite}}</h1>
```

尝试一下 »

使用 **ng-options** 指令，选择的值是一个对象：

实例
使用 **ng-options**:

```
<select ng-model="selectedSite" ng-options="x.site for x in sites">
</select>

<h1>你选择的是：{{selectedSite.site}}</h1>
<p>网址为：{{selectedSite.url}}</p>
```

尝试一下 »

当选择值是一个对象时，我们就可以获取更多信息，应用也更灵活。

数据源为对象
前面实例我们使用了数组作为数据源，以下我们将数据对象作为数据源。

```
$scope.sites = {
    site01 : "Google",
    site02 : "Runoob",
    site03 : "Taobao"
};
```

ng-options 使用对象有很大的不同，如下所示：

实例
使用对象作为数据源, **x** 为键(key), **y** 为值(value):

```
<select ng-model="selectedSite" ng-options="x for (x, y) in sites">
</select>

<h1>你选择的值是：{{selectedSite}}</h1>
```

尝试一下 »

你选择的值为在 key-**value** 对中的 **value**。
value 在 key-**value** 对中也可以是个对象：

实例
选择的值在 key-**value** 对的 **value** 中, 这是它是一个对象:

```
  $scope.cars = {
car01 : {brand : "Ford", model : "Mustang", color :   "red"},
car02 : {brand : "Fiat", model : "500", color : "white"},
  car03 : {brand : "Volvo", model : "XC90", color : "black"}
};
```

尝试一下 »

在下拉菜单也可以不使用 **key**-value 对中的 **key** , 直接使用对象的属性：

实例

```
<select ng-model="selectedCar" ng-options="y.brand for (x, y) in cars">
</select>
```

尝试一下 »

AngularJS 表格

ng-repeat 指令可以完美的显示表格。

在表格中显示数据

使用 angular 显示表格是非常简单的：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="https://cdn.bootcss.com/angular.js/1.6.3/angular.min.js"></script>
</head>
<body>

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

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("/try/angularjs/data/Customers_JSON.php")
    .then(function (result) {
        $scope.names = result.data.records;
    });
});
</script>
```

尝试一下 »

废弃声明 (v1.5)

v1.5 中`$http` 的 `success` 和 `error` 方法已废弃。使用 `then` 方法替代。

如果你使用的是 v1.5 以下版本，可以使用以下代码：

```
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("/try/angularjs/data/Customers_JSON.php")
    .success(function (response) {$scope.names = response.records;});
});
```

Customers_JSON.php 文件代码：

```
<?php
echo <<<EOT
{
"records":[
{"Name":"Alfreds Futterkiste","City":"Berlin","Country":"Germany"},
{"Name":"Ana Trujillo Emparedados y helados","City":"México D.F.","Country":"Mexico"},
{"Name":"Antonio Moreno Taquería","City":"México D.F.","Country":"Mexico"},
{"Name":"Around the Horn","City":"London","Country":"UK"},
{"Name":"B's Beverages","City":"London","Country":"UK"},
{"Name":"Berglunds snabbköp","City":"Luleå","Country":"Sweden"},
{"Name":"Blauer See Delikatessen","City":"Mannheim","Country":"Germany"},
{"Name":"Blondel père et fils","City":"Strasbourg","Country":"France"},
{"Name":"Bólido Comidas preparadas","City":"Madrid","Country":"Spain"},
{"Name":"Bon app'","City":"Marseille","Country":"France"},
{"Name":"Bottom-Dollar Marketse","City":"Tsawassen","Country":"Canada"},
{"Name":"Cactus Comidas para llevar","City":"Buenos Aires","Country":"Argentina"},
{"Name":"Centro comercial Moctezuma","City":"México D.F.","Country":"Mexico"},
{"Name":"Chop-suey Chinese","City":"Bern","Country":"Switzerland"},
{"Name":"Comércio Mineiro","City":"São Paulo","Country":"Brazil"}
]
}
EOT;
?>
```

使用 CSS 样式

为了让页面更加美观，我们可以在页面中使用CSS：

CSS 样式

```
<style>
table, th , td {
```

```
  border: 1px solid grey;
    border-collapse: collapse;
  padding: 5px;
}
table tr:nth-child(odd) {
    background-color: #f1f1f1;
}
table tr:nth-child(even) {
    background-color: #ffffff;
}
</style>
```

尝试一下 »

使用 orderBy 过滤器

排序显示，可以使用 **orderBy** 过滤器:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names | orderBy : 'Country'">
      <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
    </tr>
</table>
```

尝试一下 »

使用 uppercase 过滤器

使用 **uppercase** 过滤器转换为大写:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names">
      <td>{{ x.Name }}</td>
    <td>{{ x.Country   | uppercase }}</td>
    </tr>
</table>
```

尝试一下 »

显示序号 ($index)

表格显示序号可以在 <td> 中添加 **$index**:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names">
      <td>{{ $index + 1 }}</td>
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
    </tr>
</table>
```

尝试一下 »

使用 $even 和 $odd

AngularJS 实例

```
<table>
<tr ng-repeat="x in names">
<td ng-if="$odd"   style="background-color:#f1f1f1">{{ x.Name }}</td>
<td ng-if="$even">{{   x.Name }}</td>
<td ng-if="$odd" style="background-color:#f1f1f1">{{   x.Country }}</td>
<td ng-if="$even">{{ x.Country }}</td>
</tr>
</table>
```

尝试一下 »

AngularJS SQL

在前面章节中的代码也可以用于读取数据库中的数据。

使用 PHP 从 MySQL 中获取数据

AngularJS 实例

```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr   ng-repeat="x in names">
    <td>{{ x.Name }}</td>
```

```
      <td>{{ x.Country   }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl',   function($scope, $http) {
    $http.get("http://www.runoob.com/try/angularjs/data/Customers_MySQL.php")
      .success(function (response) {$scope.names = response.records;});
});
</script>
```

尝试一下 »

ASP.NET 中执行 SQL 获取数据

AngularJS 实例

```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr   ng-repeat="x in names">
    <td>{{ x.Name }}</td>
      <td>{{ x.Country   }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl',   function($scope, $http) {
    $http.get("http://www.runoob.com/try/angularjs/data/Customers_SQL.aspx")
    .success(function (response) {$scope.names = response.records;});
});
</script>
```

尝试一下 »

服务端代码

以下列出了几种服务端代码类型：

使用 PHP 和 MySQL。返回 JSON。

使用 PHP 和 MS Access。返回 JSON。

使用 ASP.NET, VB, 及 MS Access。 返回 JSON。

使用 ASP.NET, Razor, 及 SQL Lite。 返回 JSON。

跨域 HTTP 请求

如果你需要从不同的服务器（不同域名）上获取数据就需要使用跨域 HTTP 请求。

跨域请求在网页上非常常见。很多网页从不同服务器上载入 CSS, 图片，Js脚本等。

在现代浏览器中，为了数据的安全，所有请求被严格限制在同一域名下，如果需要调用不同站点的数据，需要通过跨域来解决。

以下的 PHP 代码运行使用的网站进行跨域访问。

```
  header("Access-Control-Allow-Origin: *");
```

更多跨域访问解决方案可参阅：PHP Ajax跨域问题最佳解决方案。

1. PHP 和 MySql 代码实例

```
  <?php
header("Access-Control-Allow-Origin: *");
header("Content-Type:   application/json; charset=UTF-8");

$conn =   new mysqli("myServer", "myUser", "myPassword", "Northwind");

$result = $conn->query("SELECT   CompanyName, City, Country FROM Customers");

$outp = "";
  while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
      if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"'   . $rs["CompanyName"] . '",';
      $outp .= '"City":"'   . $rs["City"]        . '",';
      $outp .= '"Country":"'. $rs["Country"]     . '"}';
}
$outp ='{"records":['.$outp.']}';
$conn->close();

echo($outp);
?>
```

2. PHP 和 MS Access 代码实例

```
  <?php
header("Access-Control-Allow-Origin: *");
header("Content-Type:   application/json; charset=ISO-8859-1");

$conn = new COM("ADODB.Connection");
  $conn->open("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=Northwind.mdb");

$rs = $conn->execute("SELECT CompanyName, City, Country FROM   Customers");

$outp = "";
while (!$rs->EOF) {
      if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
      $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
    $rs->MoveNext();
}
$outp ='{"records":['.$outp.']}';

  $conn->close();

echo ($outp);
?>
```

3. ASP.NET, VB 和 MS Access 代码实例

```
  <%@ Import Namespace="System.IO"%>
<%@ Import Namespace="System.Data"%>
  <%@ Import Namespace="System.Data.OleDb"%>
<%
  Response.AppendHeader("Access-Control-Allow-Origin", "*")
  Response.AppendHeader("Content-type", "application/json")
Dim conn As   OleDbConnection
Dim objAdapter As OleDbDataAdapter
Dim objTable As   DataTable
Dim objRow As DataRow
Dim objDataSet As New DataSet()
  Dim outp
Dim c
conn = New OledbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data   source=Northwind.mdb")
objAdapter = New OledbDataAdapter("SELECT   CompanyName, City, Country FROM Customers", conn)
objAdapter.Fill(objDataSet,   "myTable")
objTable=objDataSet.Tables("myTable")

outp = ""
  c = chr(34)
for each x in objTable.Rows
if outp <> "" then outp = outp & ","
  outp = outp & "{" & c & "Name"    & c & ":" & c & x("CompanyName")   & c & ","
  outp = outp &       c & "City"    & c & ":" & c   & x("City")        & c & ","
outp = outp &       c & "Country" & c   & ":" & c & x("Country")     & c & "}"
next

outp   ="{" & c & "records" & c & ":[" & outp & "]}"
response.write(outp)
conn.close
%>
```

4. ASP.NET, VB Razor 和 SQL Lite 代码实例

```
  @{
Response.AppendHeader("Access-Control-Allow-Origin", "*")
  Response.AppendHeader("Content-type", "application/json")
var db =   Database.Open("Northwind");
var query = db.Query("SELECT CompanyName,   City, Country FROM Customers");
var outp =""
var c = chr(34)
}
@foreach(var row in   query)
{
if outp <> "" then outp = outp + ","
outp = outp + "{" + c + "Name"    + c + ":" + c   + @row.CompanyName + c + ","
outp = outp +       c   + "City"    + c + ":" + c   + @row.City        + c + ","
outp = outp   +       c + "Country" + c   + ":" + c + @row.Country     + c + "}"
}
outp ="{" + c   + "records" + c + ":[" + outp + "]}"
@outp
```

AngularJS HTML DOM

AngularJS 为 HTML DOM元素的属性提供了绑定应用数据的指令。

ng-disabled 指令
ng-disabled 指令直接绑定应用程序数据到 HTML 的 disabled 属性。

AngularJS 实例

```
<div ng-app="" ng-init="mySwitch=true">

<p>
<button ng-disabled="mySwitch">点我!</button>
</p>

<p>
<input type="checkbox" ng-model="mySwitch">按钮
</p>

<p>
{{ mySwitch }}
</p>

</div>
```

尝试一下 »

实例讲解：
ng-disabled 指令绑定应用程序数据 "mySwitch" 到 HTML 的 disabled 属性。
ng-model 指令绑定 "mySwitch" 到 HTML input checkbox 元素的内容（value）。
如果 **mySwitch** 为**true**, 按钮将不可用:

```
<p>
<button disabled>点我! </button>
</p>
```

如果 **mySwitch** 为**false**, 按钮则可用:

```
<p>
<button>点我!</button>
</p>
```

ng-show 指令
ng-show 指令隐藏或显示一个 HTML 元素。

AngularJS 实例

```
<div ng-app="">

<p ng-show="true">我是可见的。</p>

<p ng-show="false">我是不可见的。</p>

</div>
```

尝试一下 »

ng-show 指令根据 **value** 的值来显示（隐藏）HTML 元素。
你可以使用表达式来计算布尔值（ true 或 false）：

AngularJS 实例

```
<div ng-app="" ng-init="hour=13">

<p ng-show="hour > 12">我是可见的。</p>

</div>
```

尝试一下 »

在下一个章节中，我们将为大家介绍更多通过点击按钮来隐藏 HTML 元素的实例。

ng-hide 指令
ng-hide 指令用于隐藏或显示 HTML 元素。

AngularJS 实例

```
<div ng-app="">

<p ng-hide="true">我是不可见的。</p>

<p ng-hide="false">我是可见的。</p>

</div>
```

尝试一下 »

AngularJS 事件

AngularJS 有自己的 HTML 事件指令。

ng-click 指令
ng-click 指令定义了 AngularJS 点击事件。

AngularJS 实例

```
<div ng-app="" ng-controller="myCtrl">

<button ng-click="count   = count + 1">点我！ </button>

<p>{{ count }}</p>

</div>
```

尝试一下 »

隐藏 HTML 元素
ng-hide 指令用于设置应用部分是否可见。
ng-hide="true" 设置 HTML 元素不可见。
ng-hide="false" 设置 HTML 元素可见。

AngularJS 实例

```
<div ng-app="myApp" ng-controller="personCtrl">

<button ng-click="toggle()">隐藏/显示</button>

<p ng-hide="myVar">
  名: <input type="text" ng-model="firstName"><br>
  姓名: <input type="text" ng-model="lastName"><br>
<br>
Full Name:   {{firstName + " " + lastName}}
</p>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl',   function($scope) {
    $scope.firstName   = "John",
      $scope.lastName = "Doe"
    $scope.myVar   = false;
    $scope.toggle = function() {
          $scope.myVar = !$scope.myVar;
    };
});
</script>
```

尝试一下 »

应用解析:
第一部分 **personController**与控制器章节类似。
应用有一个默认属性: **$scope.myVar = false;**
ng-hide 指令设置 <p>元素及两个输入域是否可见， 根据 **myVar** 的值 (true 或 false) 来设置是否可见。
toggle() 函数用于切换 **myVar** 变量的值（true 和 false）。
ng-hide="true" 让元素 **不可见**。

显示 HTML 元素
ng-show 指令可用于设置应用中的一部分是否**可见** 。
ng-show="false" 可以设置 HTML 元素 **不可见**。
ng-show="true" 可以以设置 HTML 元素可见。
以下实例使用了 ng-show 指令:

AngularJS 实例

```
<div ng-app="myApp" ng-controller="personCtrl">

<button ng-click="toggle()">隐藏/显示</button>

<p ng-show="myVar">
  名: <input type="text" ng-model="firstName"><br>
  姓: <input type="text" ng-model="lastName"><br>
<br>
姓名:   {{firstName + " " + lastName}}
</p>

</div>

<script>
```

```
var app = angular.module('myApp', []);
app.controller('personCtrl',   function($scope) {
    $scope.firstName   = "John",
    $scope.lastName = "Doe"
    $scope.myVar   = true;
    $scope.toggle = function() {
          $scope.myVar = !$scope.myVar;
    }
});
</script>
```

尝试一下 »

AngularJS 模块

模块定义了一个应用程序。
模块是应用程序中不同部分的容器。
模块是应用控制器的容器。
控制器通常属于一个模块。

创建模块
你可以通过 AngularJS 的 **angular.module** 函数来创建模块：

```
<div ng-app="myApp">...</div>

<script>

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

</script>
```

"myApp" 参数对应执行应用的 HTML 元素。
现在你可以在 AngularJS 应用中添加控制器，指令，过滤器等。

添加控制器
你可以使用 **ng-controller** 指令来添加应用的控制器:

AngularJS 实例

```
<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>
```

尝试一下 »

你可以在 AngularJS 控制器 章节学到更多关于控制器的知识。

添加指令
AngularJS 提供了很多内置的指令，你可以使用它们来为你的应用添加功能。
完整的指令内容可以参阅 AngularJS 参考手册。
此外，你可以使用模块来为你应用添加自己的指令：

AngularJS 实例

```
<div ng-app="myApp" runoob-directive></div>

<script>

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

app.directive("runoobDirective",   function() {
    return {
          template : "我在指令构造器中创建!"
    };
  });
</script>
```

尝试一下 »

你可以在 AngularJS 指令 章节学到更多关于指令的知识。

模块和控制器包含在 JS 文件中
通常 AngularJS 应用程序将模块和控制器包含在 JavaScript 文件中。

在以下实例中， "myApp.js" 包含了应用模块的定义程序， "myCtrl.js" 文件包含了控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>
<body>

<div ng-app="myApp"   ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

尝试一下 »

myApp.js

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

在模块定义中 [] 参数用于定义模块的依赖关系。
中括号[]表示该模块没有依赖，如果有依赖的话会在中括号写上依赖的模块名字。

myCtrl.js

```
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName= "Doe";
});
```

函数会影响到全局命名空间
JavaScript 中应避免使用全局函数。因为他们很容易被其他脚本文件覆盖。
AngularJS 模块让所有函数的作用域在该模块下，避免了该问题。

什么时候载入库？

在我们的实例中，所有 AngularJS 库都在 HTML 文档的头部载入。

对于 HTML 应用程序，通常建议把所有的脚本都放置在 <body> 元素的最底部。
这会提高网页加载速度，因为 HTML 加载不受制于脚本加载。
在我们的多个 AngularJS 实例中，您将看到 AngularJS 库是在文档的 <head> 区域被加载。
在我们的实例中，AngularJS 在 <head> 元素中被加载，因为对 angular.module 的调用只能在库加载完成后才能进行。
另一个解决方案是在 <body> 元素中加载 AngularJS 库，但是必须放置在您的 AngularJS 脚本前面：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>
</head>
<body>

<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>

</body>
</html>
```

尝试一下 »

AngularJS 应用

现在是时候创建一个真正的 AngularJS 单页 Web 应用（single page web application，SPA）了。

AngularJS 应用实例
您已经学习了足够多关于 AngularJS 的知识，现在可以开始创建您的第一个 AngularJS 应用程序：

我的笔记

```
<br/><br/>
<button ng-click="save()">保存</button>
<button ng-click="clear()">清除</button>
<p>
剩余字数: <b><span ng-bind="left()"
class="ng-binding">100</span></b>
</p>
</div>
<br/>
```