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 Miško 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

1 篇笔记
#1

写笔记

1、ng-app="" 定义angularJS的使用范围；
2、ng-init="变量=值;变量='值'" 初始化变量的值，有多个变量时，中间用分号隔开；
3、ng-model="变量" 定义变量名；
4、ng-bind="变量" 绑定变量名，获取该变量的数据。这里的变量就是第3条的变量名。但是一般都用双重花括号来获取变量的值，比如：{{变量}}。

小浪漫1年前 (2017-06-01)

反馈/建议

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>
```

Note 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。

Note 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 »

Note 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

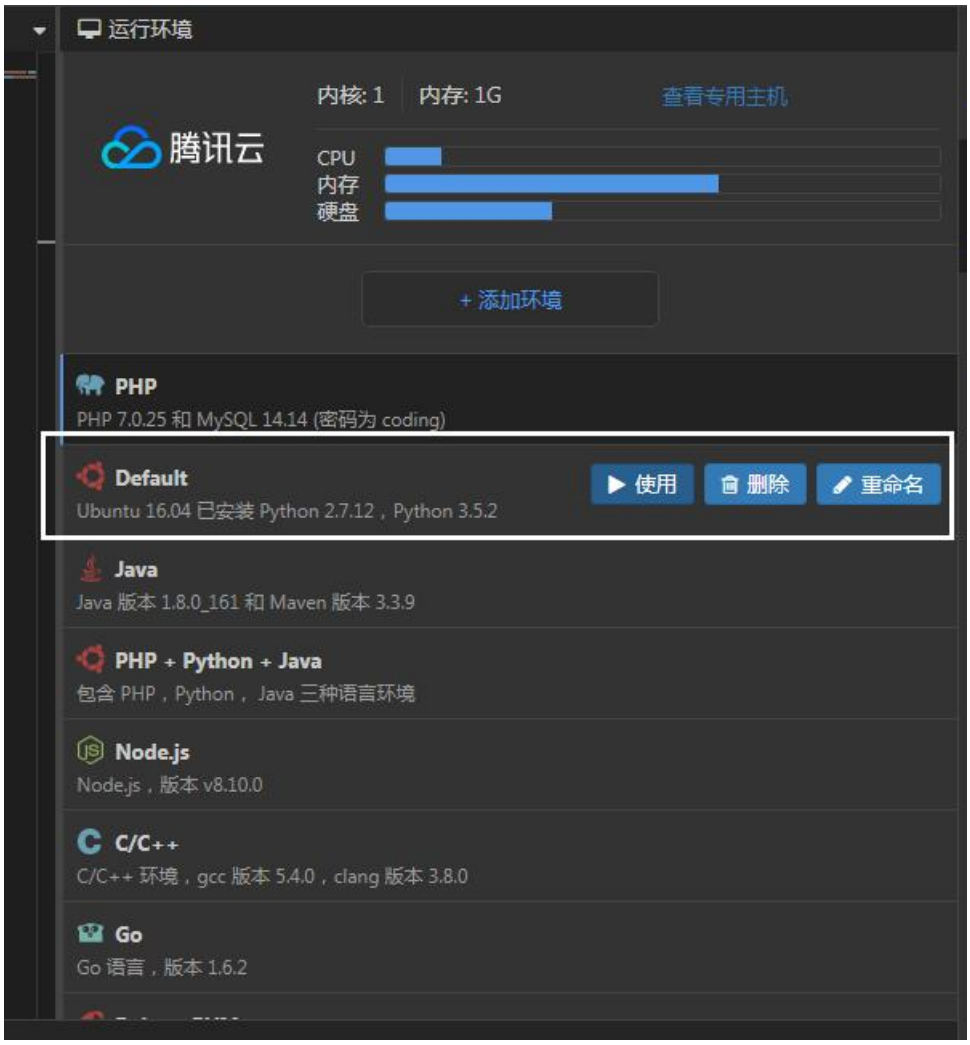
```
app.controller('myCtrl', function($scope) {
$scope.firstName= "John";
$scope.lastName= "Doe";
});
```

在接下来的教程中你将学习到更多的应用和模块的知识。

在 Cloud Studio 中使用 AngularJS

step1：访问 Cloud Studio，注册/登录账户。

step2：在右侧的运行环境菜单选择："ubuntu"



step3：在左侧代码目录中新建 html 目录，编写你的 HTML 代码，例如 index.html

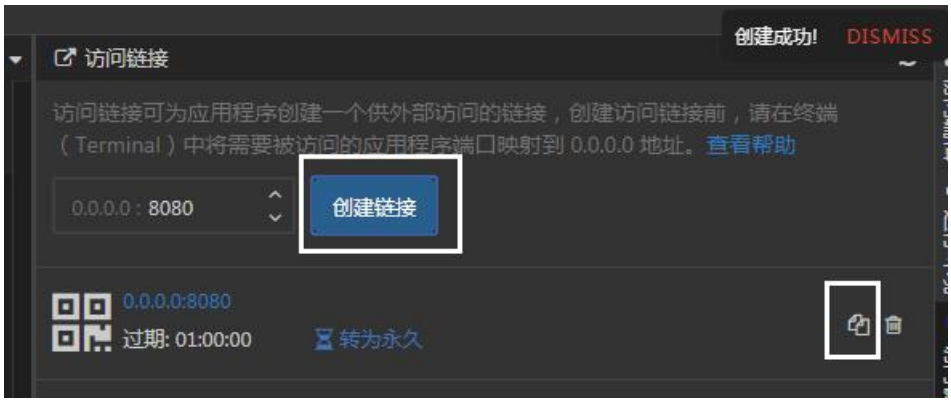
step4：在 index.html 文件的中粘贴如下代码、来引入 AngularJS：

```
<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>
```

step5：在终端中输入命令`sudo vim /etc/nginx/site-enable/default`。将配置文件红框部分修改为如下图所示，然后输入命令：`sudo nginx restart` 重启 nginx 服务（ nginx 安装完成并启动后默认会监听80端口。我们需要将 nginx 的站点目录以及监听的端口号改为我们需要的）

```
server {
        listen 8080 default_server;
        listen [::]:8080 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /home/coding/workspace/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
```

step6：点击最右侧的【访问链接】选项卡，在访问链接面板中填写端口号为：8080（和刚才 nginx 配置文件中的端口号一致），点击创建链接，即可点击生成的链接访问我们刚刚编写的代码，查看 AngularJS 效果。



AngularJS 教程

AngularJS 表达式

点我分享笔记

反馈/建议

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 »

Note 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 »

Note 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 »

Note 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 表达式 AngularJS 控制器

AngularJS API AngularJS Scope(作用域)

AngularJS ng-model 指令

ng-model 指令用于绑定应用程序数据到 HTML 控制器(input, select, textarea)的值。

ng-model 指令

ng-model 指令可以将输入域的值与 AngularJS 创建的变量绑定。

AngularJS 实例

```
<div ng-app="myApp" ng-controller="myCtrl">
    名字: <input ng-model="name">
</div>

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

```
});
</script>
```

尝试一下 »

双向绑定

双向绑定，在修改输入域的值时， AngularJS 属性的值也将修改：

AngularJS 实例

```
<div ng-app="myApp" ng-controller="myCtrl">
    名字: <input ng-model="name">
    <h1>你输入了: {{name}}</h1>
</div>
```

尝试一下 »

验证用户输入

AngularJS 实例

```
<form ng-app="" name="myForm">
    Email:
    <input type="email" name="myAddress" ng-model="text">
    <span ng-show="myForm.myAddress.$error.email">不是一个合法的邮箱地址</span>
</form>
```

尝试一下 »

以上实例中，提示信息会在 ng-show 属性返回 true 的情况下显示。

应用状态

ng-model 指令可以为应用数据提供状态值(invalid, dirty, touched, error):

AngularJS 实例

```
<form ng-app="" name="myForm" ng-init="myText = 'test@runoob.com'">
    Email:
    <input type="email" name="myAddress" ng-model="myText" required></p>
    <h1>状态</h1>
    {{myForm.myAddress.$valid}}
    {{myForm.myAddress.$dirty}}
    {{myForm.myAddress.$touched}}
</form>
```

尝试一下 »

CSS 类

ng-model 指令基于它们的状态为 HTML 元素提供了 CSS 类：

AngularJS 实例

```
<style>
input.ng-invalid {
    background-color: lightblue;
}
</style>
<body>

<form ng-app="" name="myForm">
    输入你的名字:
    <input name="myAddress" ng-model="text" required>
</form>
```

尝试一下 »

ng-model 指令根据表单域的状态添加/移除以下类：

- ng-empty
- ng-not-empty
- ng-touched
- ng-untouched
- ng-valid
- ng-invalid
- ng-dirty
- ng-pending
- ng-pristine

AngularJS API　　AngularJS Scope(作用域)

2 篇笔记

#2

写笔记

ng-invalid:未通过验证的表单
ng-valid:布尔型属性，它指示表单是否通过验证。如果表单当前通过验证，他将为true
ng-dirty:布尔值属性，表示用户是否修改了表单。如果为 ture，表示有修改过；false 表示修没有修改过
ng-touched:布尔值属性，表示用户是否和控件进行过交互
ng-pristine:布尔值属性，表示用户是否修改了表单。如果为ture，表示没有修改过；false表示修改过

luobu1年前 **(2017-07-12)**

#1

ng-valid: 验证通过
ng-invalid: 验证失败
ng-valid-[key]: 由$setValidity添加的所有验证通过的值
ng-invalid-[key]: 由$setValidity添加的所有验证失败的值
ng-pristine: 控件为初始状态
ng-dirty: 控件输入值已变更
ng-touched: 控件已失去焦点
ng-untouched: 控件未失去焦点
ng-pending: 任何为满足$asyncValidators的情况

Vivi12个月前 **(10-12)**

反馈/建议

AngularJS ng-model 指令　　AngularJS ng-app 指令

AngularJS Scope(作用域)

Scope(作用域) 是应用在 HTML (视图) 和 JavaScript (控制器)之间的纽带。

Scope 是一个对象，有可用的方法和属性。

Scope 可应用在视图和控制器上。

如何使用 Scope

当你在 AngularJS 创建控制器时，你可以将 **$scope** 对象当作一个参数传递:

AngularJS 实例

控制器中的属性对应了视图上的属性：

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

<h1>{{carname}}</h1>

</div>

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

app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
```

尝试一下 »

当在控制器中添加 **$scope** 对象时，视图 (HTML) 可以获取了这些属性。

视图中，你不需要添加 **$scope** 前缀, 只需要添加属性名即可，如： **{{carname}}**。

Scope 概述

AngularJS 应用组成如下：

View(视图), 即 HTML。

Model(模型), 当前视图中可用的数据。

Controller(控制器), 即 JavaScript 函数，可以添加或修改属性。

scope 是模型。

scope 是一个 JavaScript 对象，带有属性和方法，这些属性和方法可以在视图和控制器中使用。

AngularJS 实例

如果你修改了视图，模型和控制器也会相应更新：

```
<div ng-app="myApp" ng-controller="myCtrl">
<input ng-model="name">
<h1>{{greeting}}</h1>
<button ng-click='sayHello()'>点我</button>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.name = "Runoob";
$scope.sayHello = function() {
$scope.greeting = 'Hello ' + $scope.name + '!';
};
});
</script>
```

尝试一下 »

Scope 作用范围

了解你当前使用的 scope 是非常重要的。

在以上两个实例中，只有一个作用域 scope，所以处理起来比较简单，但在大型项目中， HTML DOM 中有多个作用域，这时你就需要知道你使用的 scope 对应的作用域是哪一个。

AngularJS 实例

当我们使用 **ng-repeat** 指令时，每个重复项都访问了当前的重复对象:

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

<ul>
    <li ng-repeat="x in names">{{x}}</li>
</ul>

</div>

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

app.controller('myCtrl', function($scope) {
    $scope.names = ["Emil", "Tobias", "Linus"];
});
</script>
```

尝试一下 »

每个 **<li>** 元素可以访问当前的重复对象，这里对应的是一个字符串， 并使用变量 **x** 表示。

根作用域

所有的应用都有一个 **$rootScope**，它可以作用在 **ng-app** 指令包含的所有 HTML 元素中。

$rootScope 可作用于整个应用中。是各个 controller 中 scope 的桥梁。用 rootscope 定义的值，可以在各个 controller 中使用。

AngularJS 实例

创建控制器时，将 $rootScope 作为参数传递，可在应用中使用：

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

<h1>{{lastname}} 家族成员:</h1>

<ul>
    <li ng-repeat="x in names">{{x}} {{lastname}}</li>
</ul>

</div>

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

app.controller('myCtrl', function($scope, $rootScope) {
    $scope.names = ["Emil", "Tobias", "Linus"];
    $rootScope.lastname = "Refsnes";
});
</script>
```

尝试一下 »

□ AngularJS ng-model 指令　　AngularJS ng-app 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 »

2 篇笔记

#2

写笔记

model 中可以有多个 controller，第一个例子可以改成这样:

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

    <div ng-controller="myCtrl1">

        名:  <input type="text" ng-model="firstName"><br>

        姓:  <input type="text" ng-model="lastName"><br>

        <br>

        姓名: {{firstName + " " + lastName}}

    </div>

    <br><br>

    <div ng-controller="myCtrl2">

        名:  <input type="text" ng-model="firstName"><br>

        姓:  <input type="text" ng-model="lastName"><br>

        <br>

        姓名: {{firstName + " " + lastName}}

    </div>

</div>
```

尝试一下 »

Jerry X7个月前 **(03-15)**

#1

关于 controller 中作用域的问题：

controller 中，如果局部 $scope 和 $rootScope 都存在，且有相同名字的变量，`{{变量名}}` 指局部变量而不是全局变量，作用域只有当前 **controller**；`{{$root.变量名}}` 是全局变量，在 **ng-app=""** 下任何一个 **controller** 中都能使用。如果没有 **$scope**, 只有 **$rootScope**，那么 `{{变量名}}` 和 `{{$root.变量名}}` 就没区别了。

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

<div ng-controller="myCtrl">       //输出结果

    {{first}}<br>                  //ctrl局部first

    {{$root.first}}<br>            //全局first

    {{second}}<br>                 //全局second

    {{$root.second}}<br>           //全局second

</div>

<br>
```

```
<br>

<div ng-controller="myCtrl2">

    {{first}}<br>                //全局first

    {{$root.first}}<br>          //全局first

    {{second}}<br>              //ctrl2局部second

    {{$root.second}}            //全局second

</div>

<script>

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

app.controller('myCtrl', function ($scope,$rootScope) {

        $scope.first = 'ctrl局部first';

        $rootScope.first = '全局first';

    });

app.controller('myCtrl2', function ($scope,$rootScope) {

        $scope.second = 'ctrl2局部second';

        $rootScope.second = '全局second';

});

</script>
```

尝试一下 »

Butters4个月前 (05-31)

反馈/建议

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 控制器　　AngularJS HTML DOM □

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 Select(选择框)　　AngularJS 动画

7 篇笔记

写笔记

反馈/建议

AngularJS 参考手册　　AngularJS 表格

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) {
$http.get("http://www.runoob.com/try/angularjs/data/sites.php")
.then(function (response) {$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。

Note

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 ng-value 指令　　AngularJS 服务(Service)

AngularJS Http　　AngularJS SQL

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 Http

AngularJS SQL

3 篇笔记

写笔记

反馈/建议

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：

1. 使用 PHP 和 MySQL。返回 JSON。
2. 使用 PHP 和 MS Access。返回 JSON。
3. 使用 ASP.NET, VB, 及 MS Access。 返回 JSON。
4. 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 表格 AngularJS 表单

点我分享笔记

反馈/建议

AngularJS 过滤器 AngularJS 事件

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 »

Note 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 HTML DOM　　AngularJS 模块 □

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", []);
```

Note 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?

Note 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 事件　　AngularJS 应用

AngularJS SQL　　AngularJS 输入验证

AngularJS 表单

AngularJS 表单是输入控件的集合。

HTML 控件

以下 HTML input 元素被称为 HTML 控件:

- input 元素
- select 元素
- button 元素

textarea 元素

数据绑定

Input 控件使用 ng-model 指令来实现数据绑定。

```
<input type="text" ng-model="firstname">
```

通过以上代码应用有了一个名为 firstname 的属性。

它通过 ng-model 指令来绑定到你的应用。

firstname 属性可以在 controller 中使用：

实例

```
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
$scope.firstname = "John";
});
```

尝试一下 »

也可以在应用的其他地方使用:

实例

```
<form>
First Name: <input type="text" ng-model="firstname">
</form>
<h1>You entered: {{firstname}}</h1>
```

尝试一下 »

Checkbox（复选框）

checkbox 的值为 true 或 false，可以使用 ng-model 指令绑定，它的值可以用于应用中：

实例

复选框选中后显示 h1 标签内容：

```
<form>
Check to show a header:
<input type="checkbox" ng-model="myVar">
</form>
<h1 ng-show="myVar">My Header</h1>
```

尝试一下 »

单选框

我们可以使用 ng-model 来绑定单选按钮到你的应用中。

单选框使用同一个 ng-model ，可以有不同的值，但只有被选中的单选按钮的值会被使用。

实例

根据选中的单选按钮，显示信息：

```
<form>
选择一个选项:
<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>
```

尝试一下 »

myVar 的值可以是 dogs, tuts, 或 cars。

下拉菜单

使用 ng-model 指令可以将下拉菜单绑定到你的应用中。

ng-model 属性的值为你在下拉菜单选中的选项：

实例

根据选中的下拉菜单选项，显示信息：

```
<form>
选择一个选项:
<select ng-model="myVar">
<option value="">
<option value="dogs">Dogs
<option value="tuts">Tutorials
<option value="cars">Cars
</select>
</form>
```

尝试一下 »

myVar 的值可以是 dogs, tuts, 或 cars。

HTML 表单

HTML 表单通常与 HTML 控件同时存在。

AngularJS 表单实例

First Name:

Last Name:

RESET

form = {{user}}

master = {{master}}

应用程序代码

```
<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>form = {{user}}</p>
<p>master = {{master}}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
$scope.master = {firstName: "John", lastName: "Doe"};
$scope.reset = function() {
$scope.user = angular.copy($scope.master);
};
$scope.reset();
});
</script>
```

尝试一下 »

Note

novalidate 属性是在 HTML5 中新增的。禁用了使用浏览器的默认验证。

实例解析

ng-app 指令定义了 AngularJS 应用。

ng-controller 指令定义了应用控制器。

ng-model 指令绑定了两个 input 元素到模型的 **user** 对象。

formCtrl 函数设置了 **master** 对象的初始值，并定义了 **reset()** 方法。

reset() 方法设置了 **user** 对象等于 **master** 对象。

ng-click 指令调用了 **reset()** 方法，且在点击按钮时调用。

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。

AngularJS SQL

AngularJS 输入验证

1 篇笔记

#1

写笔记

下拉框初始化无默认值，或者有空白选项，影响美观，可通过以下方法调整：

1.给定初始化信息（ng-init）

2.隐藏空白选项（ng-show="false"）

```
<form>

  选择一个选项:

  <select ng-model="myVar" ng-init="myVar='tuts'">

    <option ng-show="false" value="">

    <option value="dogs">Dogs

    <option value="tuts">Tutorials

    <option value="cars">Cars

  </select>

</form>
```

尝试一下 »

烈焰灬1年前 (2017-09-14)

反馈/建议

AngularJS 输入验证

AngularJS 表单和控件可以验证输入的数据。

输入验证

在前面的几个章节中，你已经学到关于 AngularJS 表单和控件的知识。

AngularJS 表单和控件可以提供验证功能，并对用户输入的非法数据进行警告。

Note 客户端的验证不能确保用户输入数据的安全，所以服务端的数据验证也是必须的。

应用代码

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

<h2>Validation Example</h2>

<form  ng-app="myApp"  ng-controller="validateCtrl"
name="myForm" novalidate>

<p>用户名:<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">用户名是必须的。</span>
  </span>
</p>

<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。</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>
```

尝试一下 »

Note HTML 表单属性 **novalidate** 用于禁用浏览器默认的验证。

实例解析

AngularJS **ng-model** 指令用于绑定输入元素到模型中。

模型对象有两个属性： **user** 和 **email**。

我们使用了 **ng-show**指令， `color:red` 在邮件的 **$dirty** 或 **$invalid** 都为 true 时才显示。

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 表单 AngularJS Bootstrap

AngularJS 包含 AngularJS ng-model 指令

AngularJS API

API 意为 **A**pplication **P**rogramming **I**nterface（应用程序编程接口）。

AngularJS 全局 API

AngularJS 全局 API 用于执行常见任务的 JavaScript 函数集合，如：

比较对象

迭代对象

转换对象

全局 API 函数使用 angular 对象进行访问。

以下列出了一些通用的 API 函数：

API	描述
angular.lowercase (<angular1.7） angular.$$lowercase()（angular1.7+）	转换字符串为小写
angular.uppercase() (<angular1.7） angular.$$uppercase()（angular1.7+）	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。

注意：自 AngularJS 1.7 之后移除 angular.lowercase 和 angular.uppercase 方法, 改为 angular.$$lowercase 和 angular.$$uppercase

angular.lowercase()

实例

```
<div ng-app="myApp" ng-controller="myCtrl">
<p>{{ x1 }}</p>
<p>{{ x2 }}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.x1 = "RUNOOB";
$scope.x2 = angular.$$lowercase($scope.x1);
});
</script>
```

尝试一下 »

angular.uppercase()

实例

```
<div ng-app="myApp" ng-controller="myCtrl">
<p>{{ x1 }}</p>
<p>{{ x2 }}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.x1 = "runoob";
$scope.x2 = angular.$$uppercase($scope.x1);
});
</script>
```

尝试一下 »

angular.isString()

实例

```
<div ng-app="myApp" ng-controller="myCtrl">
<p>{{ x1 }}</p>
<p>{{ x2 }}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.x1 = "RUNOOB";
$scope.x2 = angular.isString($scope.x1);
});
</script>
```

尝试一下 »

angular.isNumber()

实例

```
<div ng-app="myApp" ng-controller="myCtrl">
<p>{{ x1 }}</p>
<p>{{ x2 }}</p>
</div>

<script>
```

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.x1 = "RUNOOB";
$scope.x2 = angular.isNumber($scope.x1);
});
</script>
```

尝试一下 »

AngularJS 包含 AngularJS ng-model 指令

2 篇笔记

#2

写笔记

1、定义ng-app以及控制器，在输入框中填写值，x1和x2分别接收其大小写转换的值，并根据switch来判断true/false让其显示是否是字符串和数字

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

    <input type="text" ng-model="myInput" ng-blur="blur()">

    <p>输入的内容为：{{myInput}}</p>

    <p>变成小写：{{ x1 }}</p>

    <p>变成大写：{{ x2 }}</p>

  <p ng-switch = "x3">

      是不是字符串：

      <label ng-switch-when = "true">是</label>

      <label ng-switch-when = "false">不是</label>

      <label ng-switch-when = ""></label>

  </p>

     <p ng-switch = "x4">

      是不是数字：

      <label ng-switch-when = "true">是</label>

      <label ng-switch-when = "false">不是</label>

      <label ng-switch-when = ""></label>

  </p>

</div>
```

2、script内容：

```
<script>

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

app.controller('myCtrl', function($scope) {
```

```
    $scope.blur = function(){
        $scope.x1 = angular.lowercase($scope.myInput);
        $scope.x2 = angular.uppercase($scope.myInput);
        $scope.x3 = angular.isString($scope.myInput);
        // angular.isNumber 这里无效
        // $scope.x4 = angular.isNumber($scope.myInput);
        $scope.x4 = !isNaN($scope.myInput);
    }
});
</script>
```

尝试一下 »

不爱李雷的韩梅梅1年前 (2017-06-14)

#1

针对楼上说的angular.isNumber无效问题，解决可以使用如下方法：

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.blur = function(){
        if($scope.myInput ==''){
          $scope.myInput = undefined;
        }
          if(!isNaN($scope.myInput)){
          $scope.myInput = Number($scope.myInput);
        }
        $scope.x1 = angular.lowercase($scope.myInput);
        $scope.x2 = angular.uppercase($scope.myInput);
        $scope.x3 = angular.isString($scope.myInput);
    $scope.x4 = angular.isNumber($scope.myInput);
    }
});
```

尝试一下 »

ECOMA10个月前 (12-06)

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。

查看 Bootstrap教程。

Bootstrap

你可以在你的 AngularJS 应用中加入 Twitter Bootstrap，你可以在你的 <head>元素中添加如下代码:

```
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.4/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/3.3.4/css/bootstrap.min.css">
```

以下是一个完整的 HTML 实例, 使用了 AngularJS 指令和 Bootstrap 类。

HTML 代码

```
<!DOCTYPE html>
<html>
<link rel="stylesheet" href="http://apps.bdimg.com/libs/bootstrap/3.3.4/css/bootstrap.min.css">
<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>
<body ng-app="myApp" ng-controller="userCtrl">

<div class="container">

<h3>Users</h3>

<table class="table table-striped">
  <thead><tr>
    <th>Edit</th>
    <th>First Name</th>
    <th>Last Name</th>
  </tr></thead>
  <tbody><tr ng-repeat="user in users">
    <td>
      <button class="btn" ng-click="editUser(user.id)">
      <span class="glyphicon glyphicon-pencil"></span>  Edit
      </button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
  </tr></tbody>
</table>

<hr>
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
```

```
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit" placeholder="First Name">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit" placeholder="Last Name">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1" placeholder="Password">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2" placeholder="Repeat Password">
  </div>
</div>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
</div>

<script src = "myUsers.js"></script>
</body>
</html>
```

尝试一下 »

指令解析

AngularJS 指令	描述
<html ng-app	为 <html> 元素定义一个应用(未命名)
<body ng-controller	为 <body> 元素定义一个控制器
<tr ng-repeat	循环 users 对象数组，每个 user 对象放在 <tr> 元素中。
<button ng-click	当点击 <button> 元素时调用函数 editUser()
<h3 ng-show	如果 edit = true 显示 <h3> 元素
<h3 ng-hide	如果 edit = true 隐藏 <h3> 元素
<input ng-model	为应用程序绑定 <input> 元素
<button ng-disabled	如果发生错误或者 incomplete = true 禁用 <button> 元素

Bootstrap 类解析

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

JavaScript 代码

myUsers.js

```
angular.module('myApp', []).controller('userCtrl', function($scope) {
$scope.fName = '';
$scope.lName = '';
$scope.passw1 = '';
$scope.passw2 = '';
$scope.users = [
{id:1, fName:'Hege', lName:"Pege" },
{id:2, fName:'Kim',  lName:"Pim" },
{id:3, fName:'Sal',  lName:"Smith" },
{id:4, fName:'Jack', lName:"Jones" },
{id:5, fName:'John', lName:"Doe" },
{id:6, fName:'Peter',lName:"Pan" }
];
$scope.edit = true;
$scope.error = false;
$scope.incomplete = false;

$scope.editUser = function(id) {
  if (id == 'new') {
    $scope.edit = true;
    $scope.incomplete = true;
    $scope.fName = '';
    $scope.lName = '';
    } else {
    $scope.edit = false;
    $scope.fName = $scope.users[id-1].fName;
    $scope.lName = $scope.users[id-1].lName;
  }
};

$scope.$watch('passw1',function() {$scope.test();});
$scope.$watch('passw2',function() {$scope.test();});
$scope.$watch('fName', function() {$scope.test();});
$scope.$watch('lName', function() {$scope.test();});

$scope.test = function() {
```

```
  if ($scope.passw1 !== $scope.passw2) {
    $scope.error = true;
    } else {
    $scope.error = false;
  }
  $scope.incomplete = false;
  if ($scope.edit && (!$scope.fName.length ||
  !$scope.lName.length ||
  !$scope.passw1.length || !$scope.passw2.length)) {
     $scope.incomplete = true;
  }
};

});
```

JavaScript 代码解析

Scope 属性	用途
$scope.fName	模型变量 (用户名)
$scope.lName	模型变量 (用户姓)
$scope.passw1	模型变量 (用户密码 1)
$scope.passw2	模型变量 (用户密码 2)
$scope.users	模型变量 (用户的数组)
$scope.edit	当用户点击创建用户时设置为true。
$scope.error	如果 passw1 不等于 passw2 设置为 true
$scope.incomplete	如果每个字段都为空(length = 0)设置为 true
$scope.editUser	设置模型变量
$scope.watch	监控模型变量
$scope.test	验证模型变量的错误和完整性

□ AngularJS 输入验证　　AngularJS 包含 □

□

1 篇笔记

#1

□写笔记

那个 test() 的监听函数里，当如果是编辑状态时，密码和重复密码为空时，保存按钮也可以点，因为编辑状态时 `$scope.edit=false`，`$scope.incomplete=false`，因为没有判断密码和重复密码为空，所以 `$scope.error=false;`
所以这个有两种方法，一种是判断密码和重复密码为空时让 `$scope.error=true;`
第二种就是分开创建用户和编辑用户两种状态，创建用户时，所有都不能为空，编辑用户时，密码不能为空。

```
if ($scope.edit) {

        if (!$scope.fName.length || !$scope.lName.length ||!$scope.passw1.length || !$scope.passw2.length) {

        $scope.incomplete=true;

    }

}else{

    if (!$scope.passw1.length || !$scope.passw2.length) {
```

```
        $scope.incomplete=true;

    }

}
```

心动随缘6个月前 (03-23)

反馈/建议

AngularJS 包含

在 AngularJS 中，你可以在 HTML 中包含 HTML 文件。

在 HTML 中包含 HTML 文件

在 HTML 中，目前还不支持包含 HTML 文件的功能。

服务端包含

大多服务端脚本都支持包含文件功能 (**SSI**： Server Side Includes)。

使用 SSI, 你可在 HTML 中包含 HTML 文件，并发送到客户端浏览器。

PHP 实例

```
<?php require("navigation.php"); ?>
```

客户端包含

通过 JavaScript 有很多种方式可以在 HTML 中包含 HTML 文件。

通常我们使用 http 请求 (**AJAX**) 从服务端获取数据，返回的数据我们可以通过 使用 **innerHTML** 写入到 HTML 元素中。

AngularJS 包含

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

实例

```
<body ng-app="">
<div ng-include="'runoob.htm'"></div>
</body>
```

尝试一下 »

步骤如下：

runoob.htm 文件代码：

```
<h1>菜鸟教程</h1>
<p>这是一个被包含的 HTML 页面，使用 ng-include 指令来实现！</p>
```

包含 AngularJS 代码

ng-include 指令除了可以包含 HTML 文件外，还可以包含 AngularJS 代码:

sites.htm 文件代码：

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

包含的文件 "sites.htm" 中有 AngularJS 代码，它将被正常执行：

实例

```
<div ng-app="myApp" ng-controller="sitesCtrl">
<div ng-include="'sites.htm'"></div>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('sitesCtrl', function($scope, $http) {
$http.get("sites.php").then(function (response) {
$scope.names = response.data.records;
});
});
</script>
```

尝试一下 »

跨域包含

默认情况下， ng-include 指令不允许包含其他域名的文件。

如果你需要包含其他域名的文件，你需要设置域名访问白名单：

sites.htm 文件代码：

```
<body ng-app="myApp">
<div ng-include="'https://c.runoob.com/runoobtest/angular_include.php'"></div>
<script>
var app = angular.module('myApp', [])
app.config(function($sceDelegateProvider) {
$sceDelegateProvider.resourceUrlWhitelist([
'https://c.runoob.com/runoobtest/**'
]);
});
</script>
</body>
```

尝试一下 »

此外，你还需要设置服务端允许跨域访问，设置方法可参考：PHP Ajax 跨域问题最佳解决方案。

angular_include.php 文件代码：

```
<?php
// 允许所有域名可以访问
header('Access-Control-Allow-Origin:*');
echo '<b style="color:red">我是跨域的内容</b>';
?>
```

AngularJS 动画

AngularJS 提供了动画效果，可以配合 CSS 使用。

AngularJS 使用动画需要引入 angular-animate.min.js 库。

```
<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular-animate.min.js"></script>
```

还需在应用中使用模型 ngAnimate：

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

什么是动画？

动画是通过改变 HTML 元素产生的动态变化效果。

实例

勾选复选框隐藏 DIV:

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

隐藏 DIV: <input type="checkbox" ng-model="myCheck">

<div ng-hide="myCheck"></div>

</body>
```

尝试一下 »

Note 应用中动画不宜太多，但合适的使用动画可以增加页面的丰富性，也可以更易让用户理解。

如果我们应用已经设置了应用名，可以把 ngAnimate 直接添加在模型中：

实例

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

<h1>隐藏 DIV: <input type="checkbox" ng-model="myCheck"></h1>

<div ng-hide="myCheck"></div>
```

```
<script>
var app = angular.module('myApp', ['ngAnimate']);
</script>
```

尝试一下 »

ngAnimate 做了什么?

ngAnimate 模型可以添加或移除 class 。

ngAnimate 模型并不能使 HTML 元素产生动画，但是 ngAnimate 会监测事件，类似隐藏显示 HTML 元素 ，如果事件发生 ngAnimate 就会使用预定义的 class 来设置 HTML 元素的动画。

AngularJS 添加/移除 class 的指令:

- ng-show
- ng-hide
- ng-class
- ng-view
- ng-include
- ng-repeat
- ng-if
- ng-switch

ng-show 和 ng-hide 指令用于添加或移除 ng-hide class 的值。

其他指令会在进入 DOM 会添加 ng-enter 类，移除 DOM 会添加 ng-leave 属性。

当 HTML 元素位置改变时，ng-repeat 指令同样可以添加 ng-move 类 。

此外， 在动画完成后，HTML 元素的类集合将被移除。例如： ng-hide 指令会添加以下类:

- ng-animate
- ng-hide-animate
- ng-hide-add (如果元素将被隐藏)
- ng-hide-remove (如果元素将显示)
- ng-hide-add-active (如果元素将隐藏)
- ng-hide-remove-active (如果元素将显示)

使用 CSS 动画

我们可以使用 CSS transition(过渡) 或 CSS 动画让 HTML 元素产生动画效果，该部分内容你可以参阅我们的 CSS 过渡教程， CSS 动画教程。

CSS 过渡

CSS 过渡可以让我们平滑的将一个 CSS 属性值修改为另外一个：

实例

在 DIV 元素设置了 .ng-hide 类时，过渡需要花费 0.5 秒，高度从 100px 变为 0:

```
<style>
div {
    transition: all linear 0.5s;
    background-color: lightblue;
    height: 100px;
}
.ng-hide {
    height: 0;
}
</style>
```

尝试一下 »

CSS 动画

CSS 动画允许你平滑的修改 CSS 属性值:

实例

在 DIV 元素设置了 .ng-hide 类时, myChange 动画将执行，它会平滑的将高度从 100px 变为 0:

```
<style>
@keyframes myChange {
    from {
        height: 100px;
    } to {
        height: 0;
    }
}
div {
    height: 100px;
    background-color: lightblue;
}
div.ng-hide {
    animation: 0.5s myChange;
}
</style>
```

尝试一下 »

AngularJS 服务(Service)　　AngularJS 依赖注入

点我分享笔记

反馈/建议

AngularJS 动画　　AngularJS 路由

AngularJS 依赖注入

什么是依赖注入

wiki 上的解释是：依赖注入（Dependency Injection，简称DI）是一种软件设计模式，在这种模式下，一个或更多的依赖（或服务）被注入（或者通过引用传递）到一个独立的对象（或客户端）中，然后成为了该客户端状态的一部分。

该模式分离了客户端依赖本身行为的创建，这使得程序设计变得松耦合，并遵循了依赖反转和单一职责原则。与服务定位器模式形成直接对比的是，它允许客户端了解客户端如何使用该系统找到依赖

> *一句话 --- 没事你不要来找我，有事我会去找你。*

AngularJS 提供很好的依赖注入机制。以下5个核心组件用来作为依赖注入：

value

factory

service

provider

constant

value

Value 是一个简单的 javascript 对象，用于向控制器传递值（配置阶段）：

```
// 定义一个模块

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



// 创建 value 对象 "defaultInput" 并传递数据

mainApp.value("defaultInput", 5);

...



// 将 "defaultInput" 注入到控制器

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {

   $scope.number = defaultInput;

   $scope.result = CalcService.square($scope.number);



   $scope.square = function() {

      $scope.result = CalcService.square($scope.number);

   }

});
```

factory

factory 是一个函数用于返回值。在 service 和 controller 需要时创建。

通常我们使用 factory 函数来计算或返回值。

```
// 定义一个模块

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



// 创建 factory "MathService" 用于两数的乘积 provides a method multiply to return multiplication of two numbers

mainApp.factory('MathService', function() {

   var factory = {};
```

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

// 在 service 中注入 factory "MathService"
mainApp.service('CalcService', function(MathService){
   this.square = function(a) {
      return MathService.multiply(a,a);
   }
});
...
```

provider

AngularJS 中通过 provider 创建一个 service、factory等(配置阶段)。

Provider 中提供了一个 factory 方法 get()，它用于返回 value/service/factory。

```
// 定义一个模块
var mainApp = angular.module("mainApp", []);
...

// 使用 provider 创建 service 定义一个方法用于计算两数乘积
mainApp.config(function($provide) {
   $provide.provider('MathService', function() {
      this.$get = function() {
         var factory = {};

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

```
      };

   });

});
```

constant

constant(常量)用来在配置阶段传递数值，注意这个常量在配置阶段是不可用的。

```
mainApp.constant("configParam", "constant value");
```

实例

以下实例提供了以上几个依赖注入机制的演示。

AngularJS 实例 - factory

```
var mainApp = angular.module("mainApp", []);
mainApp.value("defaultInput", 5);
mainApp.factory('MathService', function() {
var factory = {};
factory.multiply = function(a, b) {
return a * b;
}
return factory;
});
mainApp.service('CalcService', function(MathService){
this.square = function(a) {
return MathService.multiply(a,a);
}
});
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
$scope.number = defaultInput;
$scope.result = CalcService.square($scope.number);
$scope.square = function() {
$scope.result = CalcService.square($scope.number);
}
});
```

尝试一下 »

AngularJS 实例 - provider

```
var mainApp = angular.module("mainApp", []);
mainApp.config(function($provide) {
$provide.provider('MathService', function() {
this.$get = function() {
var factory = {};
factory.multiply = function(a, b) {
return a * b;
}
return factory;
};
});
});
mainApp.value("defaultInput", 5);
mainApp.service('CalcService', function(MathService){
this.square = function(a) {
return MathService.multiply(a,a);
}
});
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
$scope.number = defaultInput;
```

```
$scope.result = CalcService.square($scope.number);
$scope.square = function() {
$scope.result = CalcService.square($scope.number);
}
});
```

尝试一下 »

AngularJS 动画 AngularJS 路由

2 篇笔记

#2

写笔记

1.一个对别人有依赖的东西，它想要单独测试，就需要在依赖项齐备的情况下进行。如果我们在运行时注入，就可以减少这种依赖

2.参数由定义方决定

3.与import还不完全一样

Joey2年前 (2017-03-06)

#1

factory的实例

```
<div ng-app='myApp' ng-controller='MathCtrl'>

    <input type="text" ng-model='leftNum'> x

    <input type="text" ng-model='rightNum'>

    =<span >{{multRes}}</span>

</div>
```

尝试一下 »

热爱前端1年前 (2017-09-11)

反馈/建议

AngularJS 依赖注入

AngularJS 路由

本章节我们将为大家介绍 AngularJS 路由。

AngularJS 路由允许我们通过不同的 URL 访问不同的内容。

通过 AngularJS 可以实现多视图的单页 Web 应用（single page web application，SPA）。

通常我们的 URL 形式为 **http://runoob.com/first/page**，但在单页 Web 应用中 AngularJS 通过 **#! + 标记** 实现，例如：

```
http://runoob.com/#!/first

http://runoob.com/#!/second

http://runoob.com/#!/third
```

注意 Angular1.6 之前的版本是通过 # + 标记 实现路由。

当我们点击以上的任意一个链接时，向服务端请的地址都是一样的 (http://runoob.com/)。 因为 #! 号之后的内容在向服务端请求时会被浏览器忽略掉 。 所以我们就需要在客户端实现 #! 号后面内容的功能实现。 AngularJS 路由就通过 #! + 标记 帮助我们区分不同的逻辑页面并将不同的页面绑定到对应的控制器上。

在以上图形中，我们可以看到创建了两个 URL： /ShowOrders 和 /AddNewOrder。每个 URL 都有对应的视图和控制器。

接下来我们来看一个简单的实例：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AngularJS 路由实例 - 菜鸟教程</title>
<script src="https://cdn.bootcss.com/angular.js/1.7.0/angular.min.js"></script>
<script src="https://cdn.bootcss.com/angular.js/1.7.0/angular-route.min.js"></script>
</head>
<body ng-app='routingDemoApp'>
<h2>AngularJS 路由应用</h2>
<ul>
<li><a href="#!/">首页</a></li>
<li><a href="#!/computers">电脑</a></li>
<li><a href="#!/printers">打印机</a></li>
<li><a href="#!/blabla">其他</a></li>
</ul>
<div ng-view></div>
<script>
angular.module('routingDemoApp',['ngRoute'])
.config(['$routeProvider', function($routeProvider){
$routeProvider
.when('/',{template:'这是首页页面'})
.when('/computers',{template:'这是电脑分类页面'})
.when('/printers',{template:'这是打印机页面'})
.otherwise({redirectTo:'/'});
}]);
</script>
</body>
</html>
```

尝试一下 »

实例解析：

1、载入了实现路由的 js 文件：angular-route.js。

2、包含了 ngRoute 模块作为主应用模块的依赖模块。

```
angular.module('routingDemoApp',['ngRoute'])
```

3、使用 ngView 指令。

```
<div ng-view></div>
```

该 div 内的 HTML 内容会根据路由的变化而变化。

4、配置 $routeProvider，AngularJS $routeProvider 用来定义路由规则。

```
module.config(['$routeProvider', function($routeProvider){
    $routeProvider
        .when('/',{template:'这是首页页面'})
        .when('/computers',{template:'这是电脑分类页面'})
        .when('/printers',{template:'这是打印机页面'})
        .otherwise({redirectTo:'/'});
}]);
```

AngularJS 模块的 config 函数用于配置路由规则。通过使用 configAPI，我们请求把$routeProvider注入到我们的配置函数并且使用$routeProvider.whenAPI来定义我们的路由规则。
$routeProvider 为我们提供了 when(path,object) & otherwise(object) 函数按顺序定义所有路由，函数包含两个参数:

第一个参数是 URL 或者 URL 正则规则。

第二个参数是路由配置对象。

路由设置对象

AngularJS 路由也可以通过不同的模板来实现。
$routeProvider.when 函数的第一个参数是 URL 或者 URL 正则规则，第二个参数为路由配置对象。
路由配置对象语法规则如下：

```
$routeProvider.when(url, {
    template: string,
    templateUrl: string,
    controller: string, function 或 array,
    controllerAs: string,
    redirectTo: string, function,
    resolve: object<key, function>
});
```

参数说明：

template:

如果我们只需要在 ng-view 中插入简单的 HTML 内容，则使用该参数：

```
.when('/computers',{template:'这是电脑分类页面'})
```

templateUrl:

如果我们只需要在 ng-view 中插入 HTML 模板文件，则使用该参数：

```
$routeProvider.when('/computers', {

    templateUrl: 'views/computers.html',

});
```

以上代码会从服务端获取 views/computers.html 文件内容插入到 ng-view 中。

controller:

function、string或数组类型，在当前模板上执行的controller函数，生成新的scope。

controllerAs:

string类型，为controller指定别名。

redirectTo:

重定向的地址。

resolve:

指定当前controller所依赖的其他模块。

实例

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>AngularJS 路由实例 - 菜鸟教程</title>
<script src="https://cdn.bootcss.com/angular.js/1.7.0/angular.min.js"></script>
<script src="https://cdn.bootcss.com/angular.js/1.7.0/angular-route.min.js"></script>
<script type="text/javascript">
angular.module('ngRouteExample', ['ngRoute'])
.controller('HomeController', function ($scope, $route) { $scope.$route = $route;})
.controller('AboutController', function ($scope, $route) { $scope.$route = $route;})
.config(function ($routeProvider) {
$routeProvider.
when('/home', {
templateUrl: 'embedded.home.html',
controller: 'HomeController'
}).
when('/about', {
templateUrl: 'embedded.about.html',
controller: 'AboutController'
}).
otherwise({
redirectTo: '/home'
});
});
</script>
</head>
<body ng-app="ngRouteExample" class="ng-scope">
<script type="text/ng-template" id="embedded.home.html">
<h1> Home </h1>
</script>
<script type="text/ng-template" id="embedded.about.html">
<h1> About </h1>
</script>
<div>
<div id="navigation">
<a href="#!/home">Home</a>
<a href="#!/about">About</a>
</div>
<div ng-view="">
</div>
</div>
```

```
</body>
</html>
```

尝试一下 »

AngularJS 依赖注入

1 篇笔记
#1

写笔记

路由设置对象参数规则：

```
$routeProvider.when(url,{

    template:string, //在ng-view中插入简单的html内容

    templateUrl:string, //在ng-view中插入html模版文件

    controller:string,function / array, //在当前模版上执行的controller函数

    controllerAs:string, //为controller指定别名

    redirectTo:string,function, //重定向的地址

    resolve:object<key,function> //指定当前controller所依赖的其他模块

});
```

大兵瑞恩1年前 (2017-09-30)

反馈/建议

AngularJS 模块 AngularJS 实例

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

保存 清除

剩余字数: **100**

应用程序讲解

AngularJS 实例

```
<html ng-app="myNoteApp">
<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-controller="myNoteCtrl">

<h2>我的笔记</h2>

<p><textarea ng-model="message" cols="40" rows="10"></textarea></p>

<p>
<button ng-click="save()">保存</button>
<button ng-click="clear()">清除</button>
</p>

<p>Number of characters left: <span ng-bind="left()"></span></p>

</div>

<script src="myNoteApp.js"></script>
<script src="myNoteCtrl.js"></script>

</body>
</html>
```

尝试一下 »

应用程序文件 "myNoteApp.js":

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

控制器文件 "myNoteCtrl.js":

```
app.controller("myNoteCtrl", function($scope) {
    $scope.message = "";
    $scope.left  = function() {return 100 - $scope.message.length;};
    $scope.clear = function() {$scope.message = "";};
    $scope.save  = function() {alert("Note Saved");};
});
```

<html> 元素是 AngularJS 应用: ng-app="**myNoteApp**" 的容器:

```
<html ng-app="myNoteApp">
```

 是 HTML 页面中控制器: ng-controller="**myNoteCtrl**" 的作用域:

```
<div ng-controller="myNoteCtrl">
```

ng-model 指令绑定了 <textarea> 到控制器变量 **message**:

```
<textarea ng-model="message" cols="40" rows="10"></textarea>
```

两个 **ng-click** 事件调用了控制器函数 **clear()** 和 **save()**:

```
<button ng-click="save()">Save</button>
<button ng-click="clear()">Clear</button>
```

ng-bind 指令绑定控制器函数 **left()** 到，用于显示剩余字符:

```
Number of characters left: <span ng-bind="left()"></span>
```

应用库文件需要在 AngularJs 加载后才能执行：

```
<script src="myNoteApp.js"></script>
<script src="myNoteCtrl.js"></script>
```

AngularJS 应用架构

以上实例是一个完整的 AngularJS 单页Web应用（single page web application，SPA）。

<html> 元素包含了 AngularJS 应用 (**ng-app=**)。

 元素定义了 AngularJS 控制器的作用域 (**ng-controller=**)。

在一个应用可以有很多控制器。

应用文件(**my...App.js**) 定义了应用模型代码。

一个或多个控制器文件 (**my...Ctrl.js**) 定义了控制器代码。

总结 - 它是如何工作的呢？

ng-app 指令位于应用的根元素下。

对于单页Web应用（single page web application，SPA），应用的根通常为 <html> 元素。

一个或多个 ng-controller 指令定义了应用的控制器。每个控制器有他自己的作用域：：定义的 HTML 元素。

AngularJS 在 HTML DOMContentLoaded 事件中自动开始。如果找到 ng-app 指令， AngularJS 载入指令中的模块，并将 ng-app 作为应用的根进行编译。

应用的根可以是整个页面，或者页面的一小部分，如果是一小部分会更快编译和执行。

□ AngularJS 模块 AngularJS 实例 □

□ AngularJS 应用 AngularJS 参考手册 □

AngularJS 实例

实例

您可以在线编辑实例，然后点击按钮查看结果。

AngularJS 实例

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

尝试一下 »

AngularJS 基础

我的第一个 AngularJS 表达式

我的第一个 AngularJS 指令

我的第一个 AngularJS 指令（带有有效的 HTML5）

AngularJS 控制器

基础的 AngularJS 讲解

AngularJS 表达式

简单的表达式

不使用 ng-app 的表达式

带有数字的表达式

使用带有数字的 ng-bind

带有字符串的表达式

使用带有字符串的 ng-bind

带有对象的表达式

使用带有对象的 ng-bind

带有数组的表达式

使用带有数组的 ng-bind

表达式讲解

AngularJS 指令

AngularJS 指令

ng-model 指令

ng-repeat 指令（带有数组）

ng-repeat 指令（带有对象）

指令讲解

AngularJS 控制器

AngularJS 控制器

控制器属性

控制器函数

JavaScript 文件中的控制器 I

JavaScript 文件中的控制器 II

控制器讲解

AngularJS 过滤器

表达式过滤器 uppercase

表达式过滤器 lowercase

货币过滤器表达式

指令过滤器 orderBy

输入过滤器

过滤器讲解

AngularJS XMLHttpRequest

读取静态 JSON 文件

XMLHttpRequest 解析

AngularJS 表格

显示一个表格 (简单)

带有 CSS 样式的表格

表格排序

使用 uppercase 过滤器的表格

使用序号的表格

使用 even 和 odd 表格

AngularJS - 从数据库中读取数据

从 MySQL 数据库中读取数据

从 SQL Server 数据库中读取数据

AngularJS SQL 解析

AngularJS HTML DOM

ng-disabled 指令

ng-show 指令

ng-hide 指令

HTML DOM 解析

AngularJS 事件

ng-click 指令

ng-hide 指令

ng-show 指令

HTML 事件解析

AngularJS 模块

在 body 中的 AngularJS 模块

在文件中 AngularJS 模块

AngularJS 模块解析

AngularJS 表单

AngularJS 表单

AngularJS 验证

AngularJS 表单解析

AngularJS API

AngularJS angular.lowercase()

AngularJS angular.uppercase()

AngularJS angular.isString()

AngularJS angular.isNumber()

API 解析

AngularJS Bootstrap

AngularJS 使用 Bootstrap

AngularJS 使用 Bootstrap 并使用包含文件功能

Bootstrap 解析

AngularJS 应用

AngularJS 笔记应用

AngularJS 备忘录应用

AngularJS 应用

AngularJS 应用 AngularJS 参考手册

点我分享笔记

反馈/建议

AngularJS 实例 AngularJS Http

AngularJS 参考手册

AngularJS 指令

本教程用到的 AngularJS 指令：

指令	描述
ng-app	定义应用程序的根元素。
ng-bind	绑定 HTML 元素到应用程序数据
ng-bind-html	绑定 HTML 元素的 innerHTML 到应用程序数据，并移除 HTML 字符串中危险字符
ng-bind-template	规定要使用模板替换的文本内容
ng-blur	规定 blur 事件的行为
ng-change	规定在内容改变时要执行的表达式
ng-checked	规定元素是否被选中
ng-class	指定 HTML 元素使用的 CSS 类
ng-class-even	类似 ng-class，但只在偶数行起作用
ng-class-odd	类似 ng-class，但只在奇数行起作用
ng-click	定义元素被点击时的行为
ng-cloak	在应用正要加载时防止其闪烁

ng-controller	定义应用的控制器对象
ng-copy	规定拷贝事件的行为
ng-csp	修改内容的安全策略
ng-cut	规定剪切事件的行为
ng-dblclick	规定双击事件的行为
ng-disabled	规定一个元素是否被禁用
ng-focus	规定聚焦事件的行为
ng-form	指定 HTML 表单继承控制器表单
ng-hide	隐藏或显示 HTML 元素
ng-href	为 the <a> 元素指定链接
ng-if	如果条件为 false 移除 HTML 元素
ng-include	在应用中包含 HTML 文件
ng-init	定义应用的初始化值
ng-jq	定义应用必须使用到的库，如：jQuery
ng-keydown	规定按下按键事件的行为
ng-keypress	规定按下按键事件的行为
ng-keyup	规定松开按键事件的行为
ng-list	将文本转换为列表 (数组)
ng-model	绑定 HTML 控制器的值到应用数据
ng-model-options	规定如何更新模型
ng-mousedown	规定按下鼠标按键时的行为
ng-mouseenter	规定鼠标指针穿过元素时的行为
ng-mouseleave	规定鼠标指针离开元素时的行为
ng-mousemove	规定鼠标指针在指定的元素中移动时的行为
ng-mouseover	规定鼠标指针位于元素上方时的行为
ng-mouseup	规定当在元素上松开鼠标按钮时的行为
ng-non-bindable	规定元素或子元素不能绑定数据
ng-open	指定元素的 open 属性
ng-options	在 <select> 列表中指定 <options>
ng-paste	规定粘贴事件的行为
ng-pluralize	根据本地化规则显示信息

ng-readonly	指定元素的 readonly 属性
ng-repeat	定义集合中每项数据的模板
ng-selected	指定元素的 selected 属性
ng-show	显示或隐藏 HTML 元素
ng-src	指定 <img> 元素的 src 属性
ng-srcset	指定 <img> 元素的 srcset 属性
ng-style	指定元素的 style 属性
ng-submit	规定 onsubmit 事件发生时执行的表达式
ng-switch	规定显示或隐藏子元素的条件
ng-transclude	规定填充的目标位置
ng-value	规定 input 元素的值

过滤器解析 AngularJs 过滤器。

AngularJS 事件

AngularJS 支持以下事件:

- ng-click
- ng-dbl-click
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

事件解析： Angular 事件。

AngularJS 验证属性

- $dirty
- $invalid
- $error

验证解析：Angular 验证。

AngularJS 全局 API

转换

API	描述
angular.lowercase()	将字符串转换为小写
angular.uppercase()	将字符串转换为大写

angular.copy()	数组或对象深度拷贝
angular.forEach()	对象或数组的迭代函数

比较

API	描述
angular.isArray()	如果引用的是数组返回 true
angular.isDate()	如果引用的是日期返回 true
angular.isDefined()	如果引用的已定义返回 true
angular.isElement()	如果引用的是 DOM 元素返回 true
angular.isFunction()	如果引用的是函数返回 true
angular.isNumber()	如果引用的是数字返回 true
angular.isObject()	如果引用的是对象返回 true
angular.isString()	如果引用的是字符串返回 true
angular.isUndefined()	如果引用的未定义返回 true
angular.equals()	如果两个对象相等返回 true

JSON

API	描述
angular.fromJson()	反序列化 JSON 字符串
angular.toJson()	序列化 JSON 字符串

基础

API	描述
angular.bootstrap()	手动启动 AngularJS
angular.element()	包裹着一部分DOM element或者是HTML字符串，把它作为一个jQuery元素来处理。
angular.module()	创建，注册或检索 AngularJS 模块

全局 API 解析 ： Angular API。

□ AngularJS 实例 AngularJS Http □