



AngularJS 教程

AngularJS 简介

AngularJS 表达式

AngularJS 指令

AngularJS 模型

AngularJS Scope(作用域)

AngularJS 控制器

AngularJS 过滤器

AngularJS Service

AngularJS Http

AngularJS Select

AngularJS 表格

AngularJS SQL

AngularJS HTML DOM

AngularJS 事件

AngularJS 模块

AngularJS 表单

AngularJS 输入验证

AngularJS API

AngularJS Bootstrap

[← AngularJS 表达式](#)[AngularJS 模型 →](#)

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，<div> 元素是 AngularJS 应用程序 的"所有者"。

[AngularJS 包含](#)[AngularJS 动画](#)[AngularJS 依赖注入](#)[AngularJS 路由](#)[AngularJS 应用](#)

AngularJS 实例

[AngularJS 实例](#)

AngularJS 参考手册

[AngularJS 参考手册](#)

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>
```

```
</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 表达式

AngularJS 模型 →



笔记列表



angular自定义指令的两种写法：

上面这种，感觉更清晰明确一点。



```
// angular.module('MyApp', [])
// .directive('z11', z11)
// .controller('con1', ['$scope', func1]);
//
// function z11(){
//   var directive={
//     restrict:'AEC',
//     template:'this is the it-first directive',
//   };
//   return directive;
// };
//
// function func1($scope){
//   $scope.name="alice";
// }

//这是教程里类似的写法
angular.module('myApp', []).directive('z11', [ function(){
  return {
    restrict:'AE',
    template:'thirective',
    link:function($scope,elm,attr,controller){
      console.log("这是link");
    },
    controller:function($scope,$element,$attrs){
      console.log("这是con");
    }
  };
}]).controller('Con1', ['$scope', function($scope){
  $scope.name="aliceqqq";
}]);
```

Alice 12个月前 (12-29)



还有指令配置项的：link controller等在项目运用中有遇到过：

```
angular.module('myApp', []).directive('first', [ function(){
  return {
    // scope: false, // 默认值，共享父级作用域
    // controller: function($scope, $element, $attrs, $transclude) {},
    restrict: 'AE', // E = Element, A = Attribute, C = Class, M = Comment
    template: 'first name:{{name}}',
  };
}]).directive('second', [ function(){
  return {
    scope: true, // 继承父级作用域并创建指令自己的作用域
    // controller: function($scope, $element, $attrs, $transclude) {},
    restrict: 'AE', // E = Element, A = Attribute, C = Class, M = Comment
    //当修改这里的name时，second会在自己的作用域中新建一个name变量，与父级作用域中的
    // name相对独立，所以再修改父级中的name对second中的name就不会有影响了
    template: 'second name:{{name}}',
  };
}]).directive('third', [ function(){
  return {
    scope: {}, // 创建指令自己的独立作用域，与父级毫无关系
    // controller: function($scope, $element, $attrs, $transclude) {},
    restrict: 'AE', // E = Element, A = Attribute, C = Class, M = Comment
    template: 'third name:{{name}}',
  };
}])
.controller('DirectiveController', ['$scope', function($scope){
  $scope.name="mike";
}]);
```

Alice 12个月前 (12-29)



- ng-model是用于表单元素的，支持双向绑定。对普通元素无效；
- ng-bind用于普通元素，不能用于表单元素，应用程序单向地渲染数据到元素；
- 当ng-bind和{{}}同时使用时，ng-bind绑定的值覆盖该元素的内容。



首先,非常感谢Alice提供代码!!

```
<body ng-app="myApp" ng-controller="YiMing">
姓名: <input type="text" ng-model="name"/> <br/>
    <first/>
    <second/>
<script>
angular.module('myApp', []).directive('first', [ function(){
    return {
        restrict: 'AE',
        template: 'first name:{{name}}'
    };
}]).directive('second', [ function(){
    return {
        scope: {}, // 创建指令自己的独立作用域, 与父级毫无关系
        controller: function($scope) {
            $scope.name="YiMing"
        },
        restrict: 'AE',
        template: 'second name:{{name}}'
    };
}]).controller('YiMing', ['$scope', function($scope){
    $scope.name="YiMing";
}]);
</script>
</body>
```

尝试一下 »

1. 当同时调用first和second时,只有first生效,两个template不会同时出现。
2. 当设置了 scope:{} 之后,{{name}}获取不到了,原因就是作用域的问题,想要设置这里的name的值,可以再加一个controller的字段。

奕溟 5个月前 [08-02]



借用Alice的代码继续 关于自定义的指令内容解读

```
angular.module('myApp', []).directive('first', [ function(){
    return {
        scope: false, //默认值为 false 共享父作用域 值为true时共享父级作用域并创建指令自己的


        controller: function($scope, $element, $attrs, $transclude) {}, //作用域 值为{}时创建全新的隔离作用域, 值为string时为控制器名称
        restrict: 'AE', // E = Element, A = Attribute, C = Class, M = Comment
        template: 'first name:{{name}}', //值为string、function 用于显示dom元素
        templateUrl: 'xxx.html' //值为string function 以id为xxx.html为 调用文件显示
        priority: 0 //指明指令的优先级, 若在dom上有多个指令优先级高的先执行
        replace: false // 默认值为false 当为true是直接替换指令所在的标签
        terminal: true //值为true时优先级低于此指令的其它指令无效
        link: function // 值为函数 用来定义指令行为从传入的参数中获取元素并进行处理

    };
}]).directive('second', [ function(){
    return {
        scope: true,

        // controller: function($scope, $element, $attrs, $transclude) {},
        restrict: 'AE', // E = Element, A = Attribute, C = Class, M = Comment
        //当修改这里的name时, second会在自己的作用域中新建一个name变量, 与父级作用域中的
        // name相对独立, 所以再修改父级中的name对second中的name就不会有影响了
        template: 'second name:{{name}}',
    };
}])
.controller('DirectiveController', ['$scope', function($scope){
    $scope.name="mike";
}]);
```

菜鸟 1个月前 (11-14)



 点我分享笔记

笔记需要是本篇文章的内容扩展！

[注册邀请码获取方式](#)

Copyright © 2013-2017 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1

反馈/建议