AngularJS

# 1、简单的hellword

# 2、http发起请求

## $http 是 AngularJS 中的一个核心服务，用于读取远程服务器的数据

### 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 »**](http://www.runoob.com/try/tryit.php?filename=try_ng_customers_json3)

### 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 »**](http://www.runoob.com/try/tryit.php?filename=try_ng_customers_json2)

## 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 »**](http://www.runoob.com/try/tryit.php?filename=try_ng_customers_json)

1、测试

<!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="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>  
  
</body>  
</html>



## 2、项目中使用 用来请求后端的分页数据 （我写的有点烂）

### 1、contrller

/\*\*  
 \*  
 \* @return 获取所有的联系记录并进行分页  
 \*/  
@GetMapping("listhbGetPage")  
@ResponseBody  
public Page<ContactRecord> listhb(String customerName, String pactType,Pageable pageable){  
 Page<ContactRecord> pageResult = contactService.findHbPage( customerName,pactType,pageable);  
 return pageResult;  
}

### 2、anguolarjs 代码

#### 1、按钮点击事件

<button class="btn btn-default btn-search" ng-click="getData()">查询</button>

#### 2、http请求获取数据

$scope.getData=function () {  
   
 $http({  
 method: 'GET',  
 url: '/contact/listhbGetPage',  
 params: {  
 "customerName":$("#customerName").val(),  
 "pactType":"1",  
 "page":0,  
 "size":1,  
 "sort":"addTime,desc"  
 }  
 }).then(function successCallback(response) {  
 $scope.pageResult = response.data;  
 ***console***.log(response)  
  
 }, function errorCallback(response) {  
 // 请求失败执行代码  
 });  
  
}

#### 3、数据接收绑定，注意日期格式

<table class="table margin-top-3">  
 <thead>  
 <tr>  
 <th>创建时间</th>  
 <th>主题</th>  
 <th>客户</th>  
 <th>联系人</th>  
 <th>客户类型</th>  
 <th>客户时间</th>  
 <th>联系内容</th>  
 <th>创建人</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr ng-repeat="item in pageResult.content">  
 <td ng-bind="item.addTime|date:'yyyy-MM-dd: HH:mm'"></td>  
 <td ng-bind="item.subject"></td>  
 <td ng-bind="item.customerName"></td>  
 <td ng-bind="item.connect"></td>  
 <td ng-bind="item.pactType"></td>  
 <td ng-bind="item.connectTime|date:'yyyy-MM-dd HH:mm'"></td>  
 <td ng-bind="item.content"></td>  
 <td ng-bind="item.creator"></td>  
 <td>  
 <a>编辑</a>  
 <span class="text-explode">|</span>  
 <a>删除</a>  
 </td>  
 </tr>  
 </tbody>  
</table>

#### 4、测试代码



### 3、项目中利用ng-model取值

#### 1、确定项目地址

<#assign ctx=Request.request.getContextPath()>  
<link rel="icon" href="https://admore-cas.oss-cn-hangzhou.aliyuncs.com/assets/img/common/favicon.png" type="image/x-icon">  
<link rel="shortcut icon" href="https://admore-cas.oss-cn-hangzhou.aliyuncs.com/assets/img/common/favicon\_green.png" type="image/x-icon">  
<meta charset="utf-8">  
<meta name="renderer" content="webkit|ie-stand">  
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  
  
<script type="text/javascript">  
 var ***ctx*** = ***window***.ctx = "${ctx}";  
 var ***oss\_host*** = "${oss\_host}";  
</script>

#### 2、http请求

$scope.getData=function (page) {  
 $scope.query.page = page;  
 $http.post(***ctx*** + '/contact/listhbGetPage',$scope.query)  
 .success(function(data, status) {  
 $scope.pageResult = data;  
 })  
 .error(function(data, status) {  
 ngDialog.alert('查询失败');  
 });  
}

#### 3、查询条件的值得设置

<div class="flex-wrap">  
 <div class="dropdown adDropdown">  
 <input type="text" ng-model="query.customerName" class="condition" neme="customerName" placeholder="客户信息查询">  
 </div>  
 <div class="dropdown adDropdown">  
 <button class="btn btn-default dropdown-toggle multi-select flex-around-center" style="width: 130px;" type="button" id="dropdownMenu1" data-toggle="dropdown" aria-haspopup="true" aria-expanded="true">  
 {{typeName}}  
 <span class="caret"></span>  
 </button>  
 <ul class="dropdown-menu" aria-labelledby="dropdownMenu1">  
 <li ng-repeat="item in typeList"><a ng-click="chooseType(item)">{{***item***.label}}</a></li>  
 </ul>  
 </div>  
 <button class="btn btn-default btn-search" ng-click="getData(0)">查询</button>  
</div>  
<button class="btn-primary pull-right" ng-click="newContact()">新建</button>

#### 4、3中的item赋值 ，第一个不给值设置为null即可，后天有对比接收

$scope.typeList=[  
 {id:null,label:'----'},  
 {id:1,label:'实地拜访'},  
 {id:2,label:'电话联系'},  
 {id:3,label:'邮件'},  
 {id:4,label:'微信'}  
];

$scope.typeName=$scope.typeList[0].label;

$scope.chooseType=function (item) {  
 $scope.query.pactType = item.id;  
 $scope.typeName=item.label;  
}

#### 5、首页进入list地址 http://localhost:8080/contact#/list

$scope.getData(0);

#### 6、完整js代码

var ***ContactController*** = ***app***.controller('ContactController', function($rootScope,$scope,$http,$location,ngDialog) {  
  
 // console.log(1)  
 /\*\*  
 \* 初始化数据  
 \*/  
 $scope.pageResult = {};  
 $scope.query = {};  
  
 $scope.typeList=[  
 {id:null,label:'----'},  
 {id:1,label:'实地拜访'},  
 {id:2,label:'电话联系'},  
 {id:3,label:'邮件'},  
 {id:4,label:'微信'}  
 ];  
$scope.typeName=$scope.typeList[0].label;  
  
  
  
  
 $scope.getData=function (page) {  
 $scope.query.page = page;  
 $http.post(***ctx*** + '/contact/listhbGetPage',$scope.query)  
 .success(function(data, status) {  
 $scope.pageResult = data;  
 })  
 .error(function(data, status) {  
 ngDialog.alert('查询失败');  
 });  
 }  
  
  
  
 $scope.chooseType=function (item) {  
 $scope.query.pactType = item.id;  
 $scope.typeName=item.label;  
 }  
  
 $scope.newContact=function () {  
 ***window***.location.href="/contact#/new";  
 };  
  
 $scope.getData(0);  
  
  
});

### 4、改进上面的数据字典，采用后台的方式

#### 1、controller

@RequestMapping("map")  
@ResponseBody  
public Map map(){  
 Map<String,String> map = new HashMap<>();  
 map.put( "1","实地拜访");  
 map.put( "2","电话联系");  
 map.put( "3","邮件");  
 map.put( "4","微信");  
 return map;  
}

#### 2、请求赋值

*/\* $scope.typeList=[  
 {id:null,label:'----'},  
 {id:1,label:'实地拜访'},  
 {id:2,label:'电话联系'},  
 {id:3,label:'邮件'},  
 {id:4,label:'微信'}  
 ];\*/*// $scope.typeName=$scope.typeList[0].value;

$http.get(***ctx*** + '/contact/map')  
 .success(function(data, status) {  
 $scope.typeList = data;  
 })  
 .error(function(data, status) {  
 ngDialog.alert('查询失败');  
 });

#### 3、页面map绑定

*<#--<li ng-repeat="item in typeList"><a ng-click="chooseType(item)">{{item.label}}</a></li>-->*<li ng-repeat="(key,value) in typeList"><a ng-click="chooseType(key,value)">{{***value***}}</a></li>

#### 4、点击变化key值，

$scope.chooseType=function (key,value) {  
 $scope.query.pactType = key;  
 $scope.typeName=value;  
}

## 5、参数传递详解

$scope.deleteById=function (id) {  
  
 $http.get(***ctx*** + '/contact/deleteById',{  
 params: {  
 "id":id  
 }}).success(function(data, status) {  
 ngDialog.alert("删除成功"+status);  
 ***console***.log("删除成功"+data)  
 })  
 .error(function(data, status) {  
 ngDialog.alert('删除失败'+status);  
 ***console***.log("删除失败"+data)  
  
 });  
}

$http.get请求数据的格式

**[js]** [view plain](http://blog.csdn.net/hbiao68/article/details/52683348) [copy](http://blog.csdn.net/hbiao68/article/details/52683348)

1. $http.get(URL,{
2. params: {
3. "id":id
4. }
5. })
6. .success(**function**(response, status, headers, config){
8. })

$http.post请求数据的格式

**[js]** [view plain](http://blog.csdn.net/hbiao68/article/details/52683348) [copy](http://blog.csdn.net/hbiao68/article/details/52683348)

1. $http.post(URL,{
2. "id":id
3. })
4. .success(**function**(response, status, headers, config){
6. })

tips:

get 和 post方法传递参数的方式不一样。

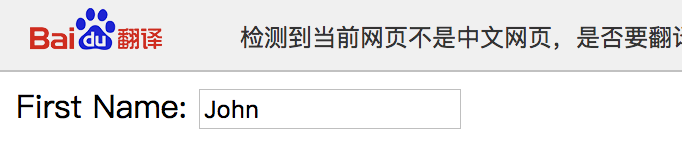
**[java]** [view plain](http://blog.csdn.net/hbiao68/article/details/52683348) [copy](http://blog.csdn.net/hbiao68/article/details/52683348)

1. //分享商品 list 列表
2. .factory("shareGoodsListService",["$http",function($http){
3. **return** {
4. //得到所有的收货地址
5. updateAddress : function(\_\_scope\_\_,addrid){
7. //获取地址的值
8. var Province = $("select[name='Province']").val();
9. var City = $("select[name='City']").val();
10. var Area = $("select[name='Area']").val();
12. var formData = {
13. id:addrid,
14. name:\_\_scope\_\_.formData.name,
15. mobile:\_\_scope\_\_.formData.mobile,
16. province:Province,
17. city:City,
18. area:Area,
19. address:\_\_scope\_\_.formData.address,
20. zip:\_\_scope\_\_.formData.zip,
21. submit:"submit"
22. };
24. $http.post("{:U('AddressInfo/editAddress')}",formData).success(function(response, status, headers, config){
25. **if**(response.status == 1){
26. alert("修改成功！");
27. }
28. })
29. },
30. getAddress:function(){
31. //$http.get方法 传递参数使用的是 {params:jsonObj}
32. $http.get("{:U('AddressInfo/editAddress')}",{
33. params: {
34. "id":id
35. }
36. })
37. .success(function(response, status, headers, config){
38. //获取地址的数据
39. **if**(response.status == 1){
40. \_\_scope\_\_.formData = {
41. name:response.data.name,
42. mobile:response.data.mobile,
43. address:response.data.address,
44. zip:response.data.zip
45. };
46. //默认收货地址的值
47. **new** PCAS("Province","City","Area",response.data.province,response.data.city,response.data.area);
48. }
49. })
50. }
52. }
53. }])

# 3、AngularJS 表单

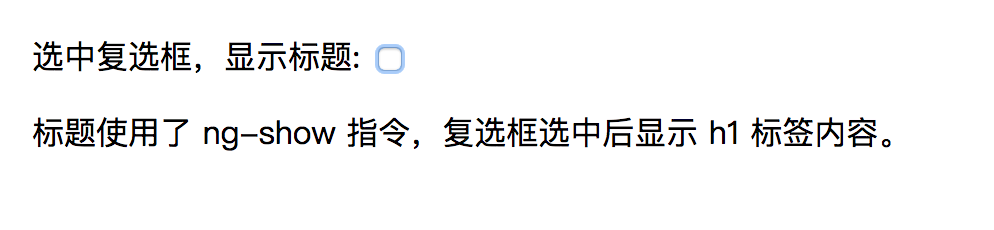
## 1、input数据 给定初始值，绑定

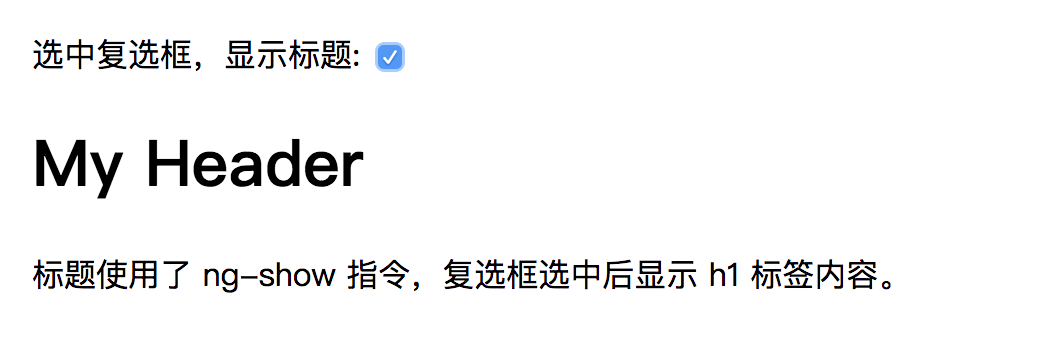
<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <script src="https://cdn.bootcss.com/angular.js/1.4.6/angular.min.js"></script>  
</head>  
<body>  
  
<div ng-app="myApp" ng-controller="formCtrl">  
 <form>  
 First Name: <input type="text" ng-model="firstname">  
 </form>  
</div>  
  
<script>  
 var ***app*** = angular.module('myApp', []);  
 ***app***.controller('formCtrl', function($scope) {  
 $scope.firstname = "John";  
 });  
</script>  
  
</body>  
</html>



## 2、Checkbox（复选框）ng-show=

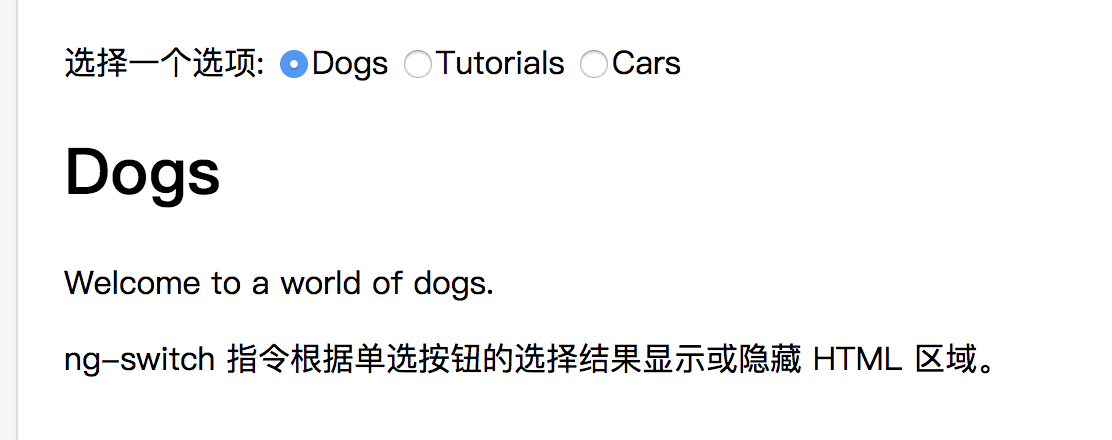
<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <script src="https://cdn.bootcss.com/angular.js/1.4.6/angular.min.js"></script>  
</head>  
<body>  
  
<div ng-app="">  
 <form>  
 选中复选框，显示标题:  
 <input type="checkbox" ng-model="myVar">  
 </form>  
 <h1 ng-show="myVar">My Header</h1>  
</div>  
  
<p>标题使用了 ng-show 指令，复选框选中后显示 h1 标签内容。</p>  
  
  
</body>  
</html>

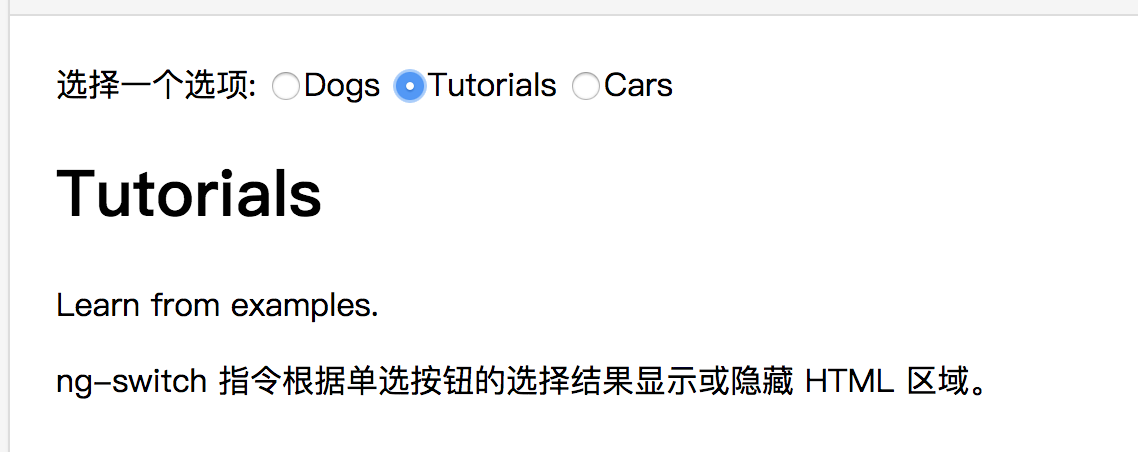




## 3、radio 单选框 （ng-switch ,ng-switch-when=）

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <script src="https://cdn.bootcss.com/angular.js/1.4.6/angular.min.js"></script>  
</head>  
<body ng-app="">  
  
<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>  
  
<div ng-switch="myVar">  
 <div ng-switch-when="dogs">  
 <h1>Dogs</h1>  
 <p>Welcome to a world of dogs.</p>  
 </div>  
 <div ng-switch-when="tuts">  
 <h1>Tutorials</h1>  
 <p>Learn from examples.</p>  
 </div>  
 <div ng-switch-when="cars">  
 <h1>Cars</h1>  
 <p>Read about cars.</p>  
 </div>  
</div>  
  
<p>ng-switch 指令根据单选按钮的选择结果显示或隐藏 HTML 区域。</p>



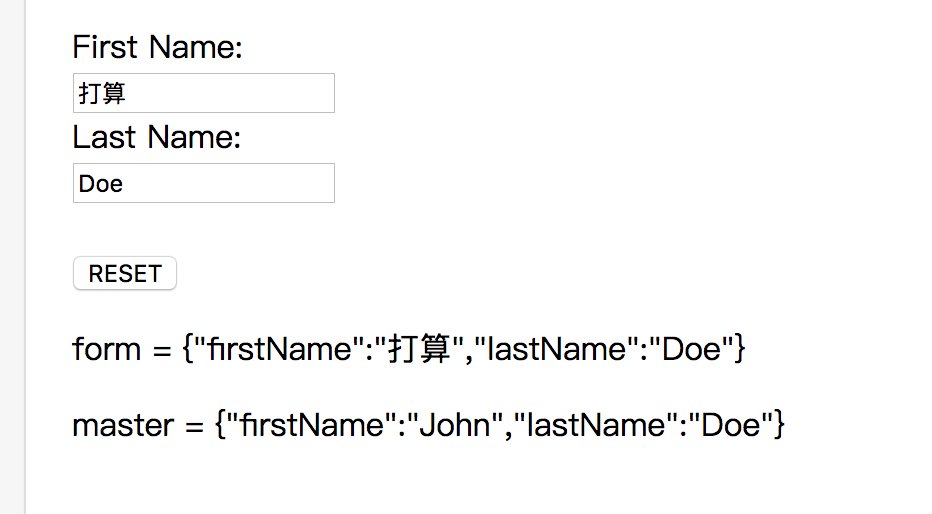


## 4、select 下拉框 ,其实和上面的是一样的

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <script src="https://cdn.bootcss.com/angular.js/1.4.6/angular.min.js"></script>  
</head>  
<body ng-app="">  
  
<form>  
 选择一个选项:  
 <select ng-model="myVar"> <!-- 给定初始值、；ng-init="myVar='tuts'" -->  
 <option value="">  
 <option value="dogs">Dogs  
 <option value="tuts">Tutorials  
 <option value="cars">Cars  
 </select>  
</form>  
  
<div ng-switch="myVar">  
 <div ng-switch-when="dogs">  
 <h1>Dogs</h1>  
 <p>Welcome to a world of dogs.</p>  
 </div>  
 <div ng-switch-when="tuts">  
 <h1>Tutorials</h1>  
 <p>Learn from examples.</p>  
 </div>  
 <div ng-switch-when="cars">  
 <h1>Cars</h1>  
 <p>Read about cars.</p>  
 </div>  
</div>  
  
<p>ng-switch 指令根据下拉菜单的选择结果显示或隐藏 HTML 区域。</p>  
  
</body>  
</html>

# 4、点击事件 （ng-click=）

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <script src="https://cdn.bootcss.com/angular.js/1.4.6/angular.min.js"></script>  
</head>  
<body>  
  
<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>  
  
</body>  
</html>



# 5、初始化值和绑定值 （ng-init，ng-bind）

<!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="" ng-init="firstName='John'">  
  
 <p>姓名为 <span ng-bind="firstName"></span></p>  
  
</div>  
  
</body>  
</html>



# 6、创建自定义的指令

## 1、解释、：html中的标签，采用驼峰的方法

## 2.、举例

1、<!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 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>  
</html>

## 3、限制使用

你可以通过以下方式来调用指令：

| **属性值** | **调用方式** | **示例** |
| --- | --- | --- |
| A （Attribute首字母） | 属性名 | <div my-directive></div> |
| C （Class 首字母） | 类名 | <div class='my-directive'></div> |
| E （Element 首字母） | 元素名 | <my-directive></my-directive> |
| M | 注释 | <!-- 指令: my-directive> |

### 1、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 »**](http://www.runoob.com/try/try.php?filename=try_ng_directive_element)

属性

<div runoob-directive></div>

[**尝试一下 »**](http://www.runoob.com/try/try.php?filename=try_ng_directive_attribute)

类名

<div class="runoob-directive"></div>

[**尝试一下 »**](http://www.runoob.com/try/try.php?filename=try_ng_directive_class)

注释

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

[**尝试一下 »**](http://www.runoob.com/try/try.php?filename=try_ng_directive_comment)

你可以限制你的指令只能通过特定的方式来调用。

### 2、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 »**](http://www.runoob.com/try/try.php?filename=try_ng_directive_restrictions)

**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。

## ****4、自定义指令属性详解****

| **属性** | **值类型** | **说明** |
| --- | --- | --- |
| restrict | string | 指令的调用方式，A、C、E、M |
| priority | number | 指令执行的优先级 |
| template | string | 指令使用的模板，可将html页面代码写于此。只能与templateUrl二选其一 |
| templateUrl | string | 从指定的url地址加载模板。只能与template二选其一 |
| replace | boolean | 是否用模板替换当前元素。true : 将指令标签替换成temple中定义的内容,页面上不会再有<my-directive>标签；false ：则append（追加）在当前元素上，即模板的内容包在<my-directive>标签内部。默认false。 |
| transclude | boolean | 是否将当前元素的内容转移到模板中 |
| scope | boolean /object | 指定指令的作用域。false(默认值): 使用父作用域作为自己的作用域（每个引用自定义指令的标签若其中一个标签改变某一变量值，则会影响其他标签的值 ）。true: 新建一个作用域，该作用域继承父作用域（两个引用自定义指令的标签之间的变量互不影响）。JavaScript对象：与父作用域隔离，并指定可以从父作用域访问的变量 |
| controller | function | 定义与其他指令进行交互的接口函数 |
| require | string | 指定需要依赖的其他指令 |
| link | function | 以编程的方式操作DOM，包括添加监听器等 |
| compile | function | 编程的方式修改DOM模板的副本，可以返回链接函数 |

### 1、****.templateUrl****

如果template里拼写的html页面代码十分的多页复杂，拼字符串的话就太麻烦啦，这里我们就可以选择templateUrl。我们可以将要拼写的html页面代码独立到一个页面里，如template.html；然后再指定该html文件所在的路径即可，如templateUrl:”template.html”。用到该自定义指令时，会自动发一个http请求来获取template.html对应的模板内容。这样做的缺点是，多了一个http请求。别急，可以改进的：   
angularJS规定了模板还可以用<Script>标签定义：

<script type="text/ng-template" id="template.html">  
 <div>自定义指令模板用Script标签定义的方式，须放在html页面ng-controller指令所在标签的内部</div>  
</script>

上面代码写在html页面的ng-controller指令所在标签的里面，这样就不用再去请求它了。示例：

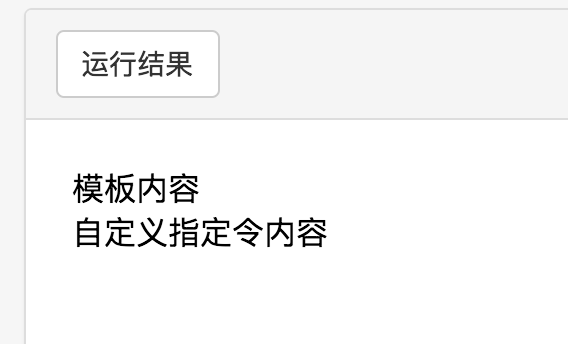
<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular.min.js"></script>  
</head>  
<body >  
<div ng-app="myApp" ng-controller="myController">  
  
 <!-- 引用自定义指令 -->  
 <my-directive></my-directive>  
  
 <!-- 模板代码：须放在myController所在标签内部 -->  
 <script type="text/ng-template" id="template.html">  
 <div> 自定义指令模板的templateUrl形式</div>  
 </script>  
</div>  
  
<script>  
 //创建模块  
 var ***app*** = angular.module('myApp', []);  
 //创建控制器  
 ***app***.controller('myController', function($scope) { });  
 //创建自定义指令  
 ***app***.directive("myDirective", function() {  
 return {  
 restrict:'E',  
 templateUrl : "template.html"  
 };  
 });  
</script>  
</body>  
  
</html>

有多个模板时，我们可以将所有的模板集中在一个文件中，只需加载一次，然后根据id的不同调用不同的模板。

### ****2、transclude****

定义是否将当前元素(html页面的自定义指令)的内容转移到模板中。 模板中要接收当前元素内容的标签需要使用ng-transclude指令。

<body >  
<div ng-app="myApp" ng-controller="myController">  
 <!-- 引用自定义指令 -->  
 <my-directive>自定义指定令内容</my-directive>  
  
 <!-- 模板代码 -->  
 <script type="text/ng-template" id="template.html">  
 <div> 模板内容</div>  
 <div ng-transclude></div>//模板接收上面自定义指令间的内容  
 </script>  
</div>  
  
<script>  
 //创建模块  
 var ***app*** = angular.module('myApp', []);  
 //创建控制器  
 ***app***.controller('myController', function($scope) { });  
 //创建自定义指令  
 ***app***.directive("myDirective", function() {  
 return {  
 templateUrl : "template.html",  
 transclude : true//转移到模板中  
 };  
 });  
</script>  
</body>

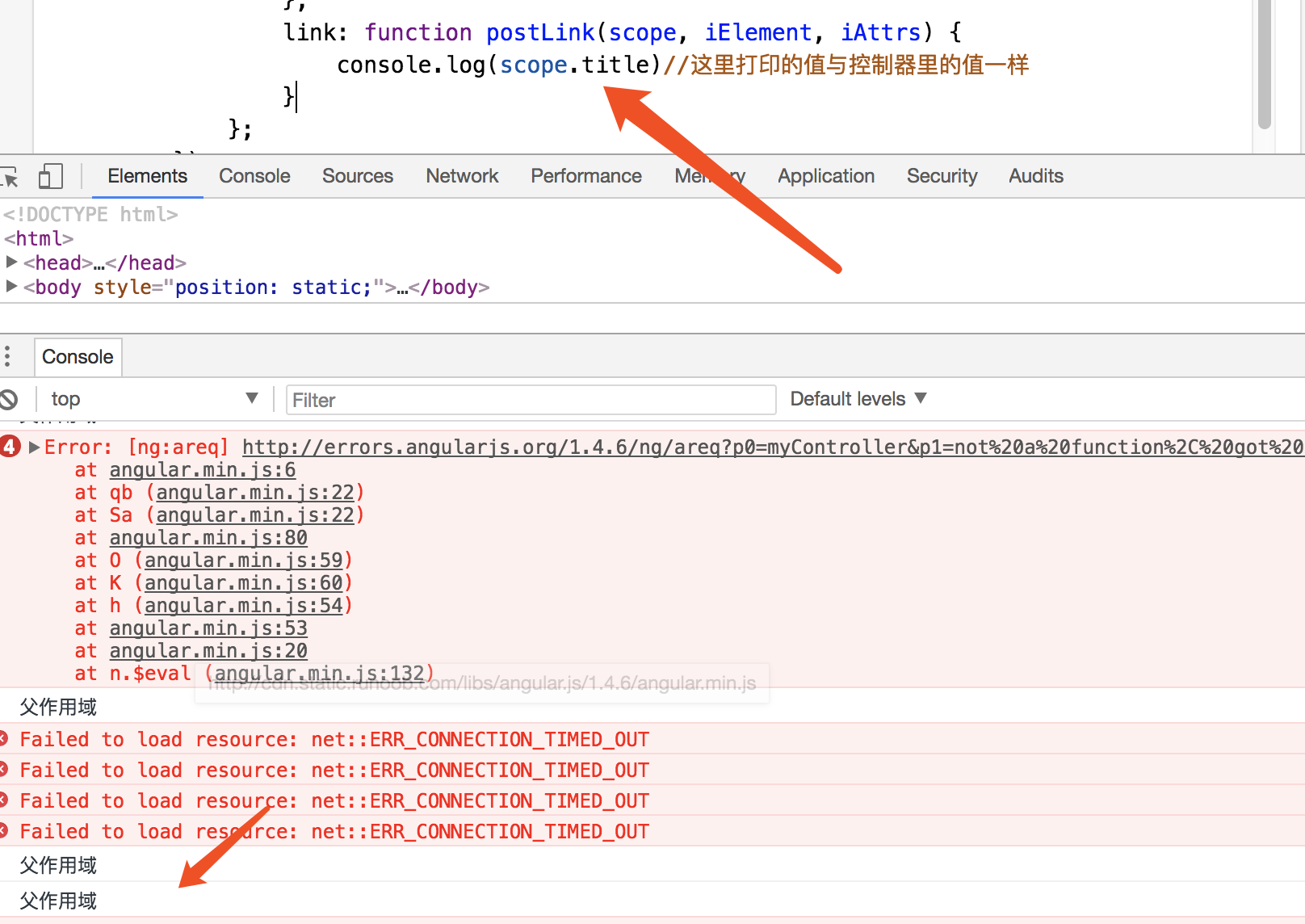


## 3、****什么是scope的父作用域****

引用自定义指令的html页面的控制器所能控制的范围。下面代码的父作用域就是**myController**所控制的范围

<!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>  
  
<!-- =符号的用法-->  
<body >  
<div ng-app="myApp" ng-controller="myController">  
 <!-- 引用自定义指令：obj变量与控制器里的objc变量双向绑定了值 -->  
 <my-directive speak="obj"></my-directive>  
</div>  
  
<script>  
 //创建模块  
 var ***app*** = angular.module('myApp', []);  
  
 //创建控制器  
 ***app***.controller('myController', function($scope) {  
 $scope.obj="父作用域";//父作用域给自定义指令属性赋的值  
 });  
  
 //创建自定义指令  
 ***app***.directive("myDirective", function() {  
 return {  
 template : "<p>模板内容</p>",  
 scope:{  
 title:"=speak"//定义一个speak属性供html页面的自定义指令用。如果写成title:"="格式，则自定义指令里的属性名就是title。  
 },  
 link: function postLink(scope, iElement, iAttrs) {  
 ***console***.log(scope.title)//这里打印的值与控制器里的值一样  
 }  
 };  
 });  
</script>  
</body>  
  
</html>

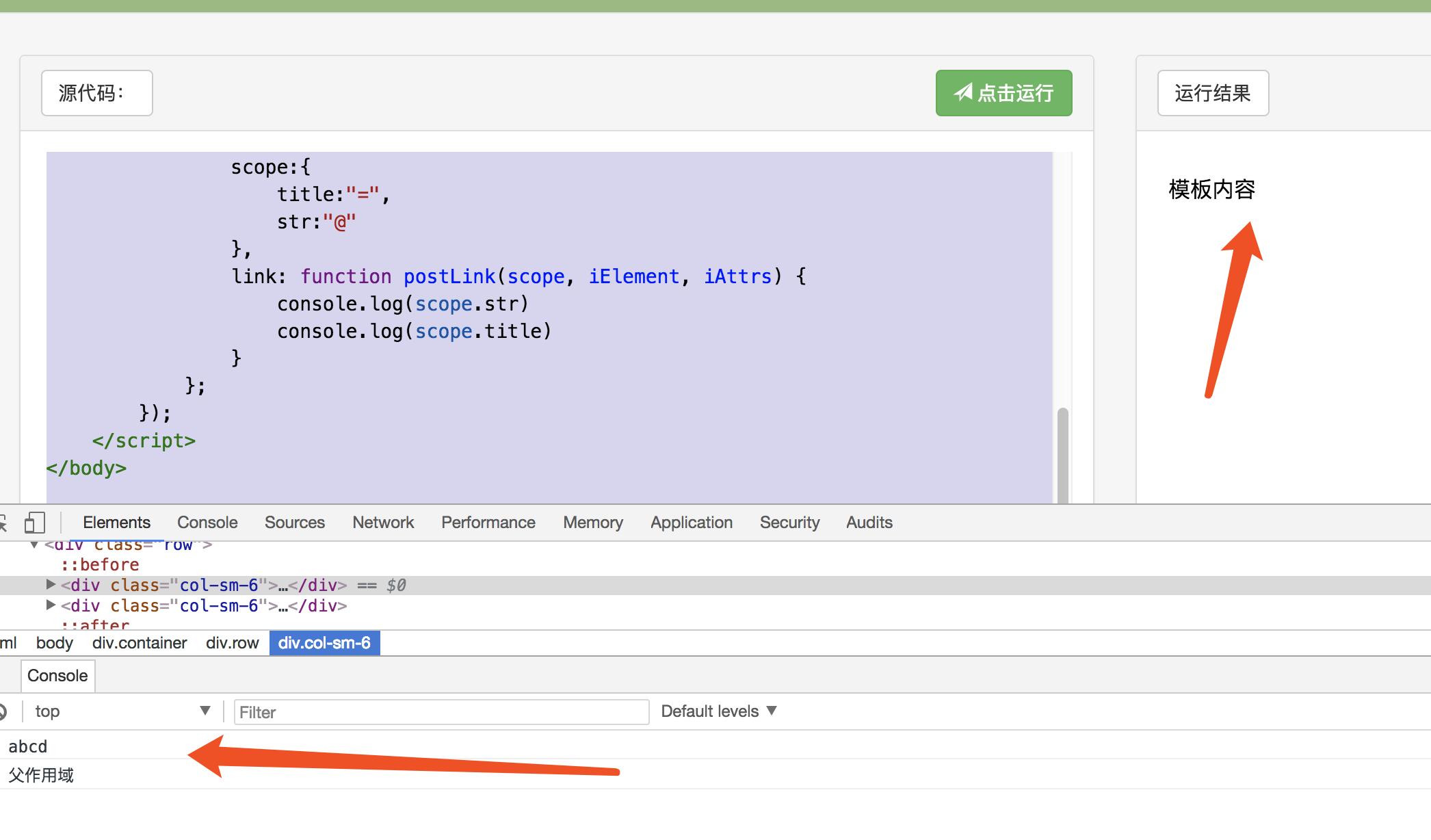




### 2、符号的使用 =、@符号的用法

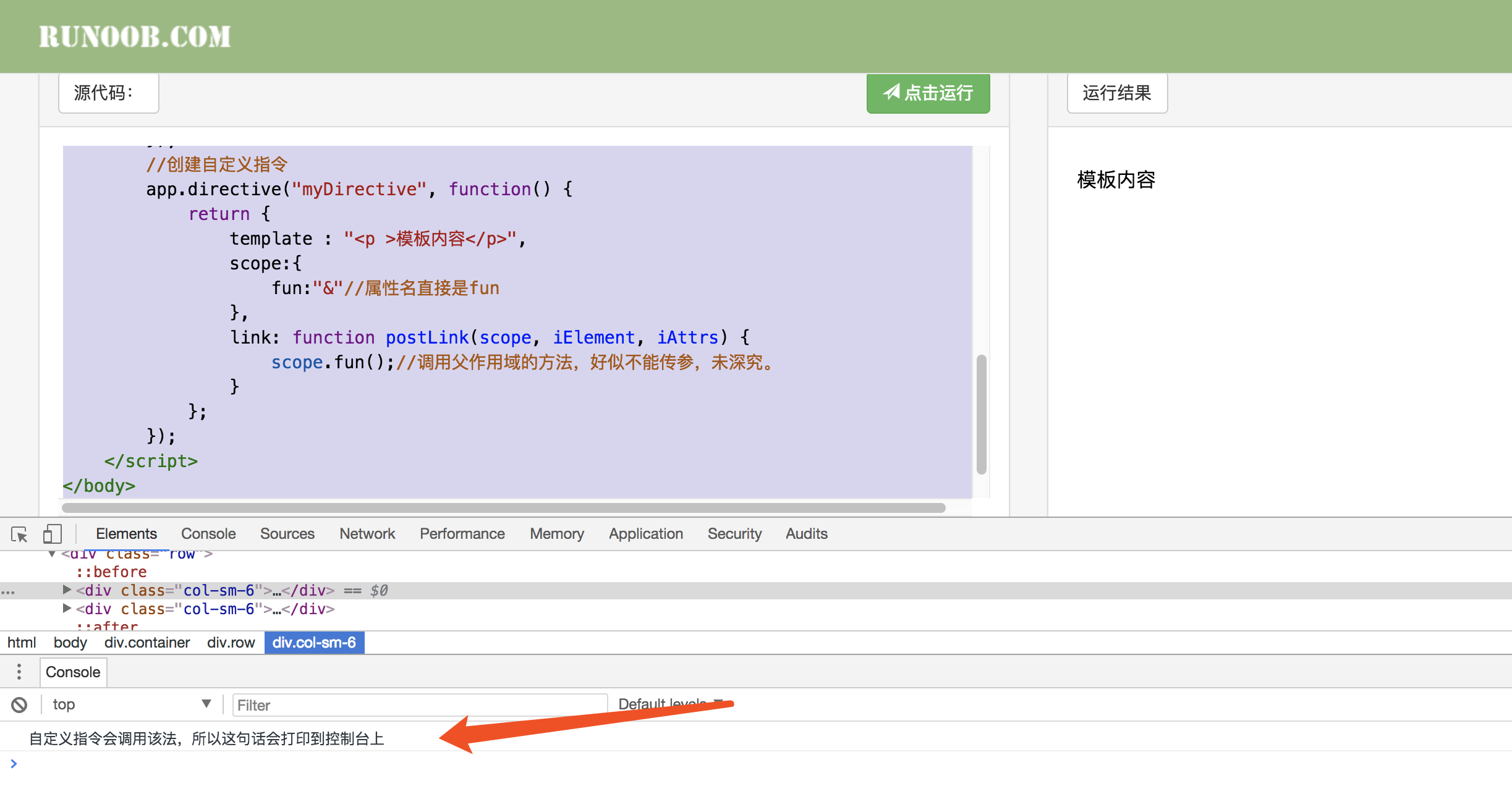
| **符号** | **说明** | **示例** |
| --- | --- | --- |
| @ | 值传递，单向绑定。html页面自定义指令里的val属性的值可传给link的scope使用。第一种写法——str : “@”，这种写法html页面的指令属性名为str | str : “@val”,属性名为val |
| = | 双向绑定数据到指令的属性中，数据值可以是任意类型的。第一种写法：name : “=”，这种写法html页面的自定义指令属性名就是name | name : “=username”,属性名是username |
| & | 使用父作用域中的一个函数,可以在指令中调用。第一种写法：getName:”&”,这种写法html页面的自定义指令属性名就是gegName | getName : “&getUserName”,属性名是getUserName |

<!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>  
  
<!-- =符号的用法-->  
<!-- @符号的用法 -->  
<body >  
<div ng-app="myApp" ng-controller="myController">  
 <!-- 引用自定义指令 -->  
 <my-directive title="obj" str="abcd">自定义指定令的内容555</my-directive>  
</div>  
  
<script>  
 //创建模块  
 var ***app*** = angular.module('myApp', []);  
 //创建控制器  
 ***app***.controller('myController', function($scope) {  
 $scope.obj="父作用域";//父作用域给自定义指令属性赋的值  
 });  
 //创建自定义指令  
 ***app***.directive("myDirective", function() {  
 return {  
 template : "<p >模板内容</p>",  
 scope:{  
 title:"=",  
 str:"@"  
 },  
 link: function postLink(scope, iElement, iAttrs) {  
 ***console***.log(scope.str)  
 ***console***.log(scope.title)  
 }  
 };  
 });  
</script>  
</body>  
  
</html>



### 3、$的使用

<!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>  
  
<!-- &符号的用法 -->  
<body >  
<div ng-app="myApp" ng-controller="myController">  
 <!-- 引用自定义指令 -->  
 <my-directive fun="test()"></my-directive>  
</div>  
  
<script>  
 //创建模块  
 var ***app*** = angular.module('myApp', []);  
 //创建控制器  
 ***app***.controller('myController', function($scope) {  
 $scope.test = function(){  
 ***console***.log('自定义指令会调用该法，所以这句话会打印到控制台上')  
 }  
 });  
 //创建自定义指令  
 ***app***.directive("myDirective", function() {  
 return {  
 template : "<p >模板内容</p>",  
 scope:{  
 fun:"&"//属性名直接是fun  
 },  
 link: function postLink(scope, iElement, iAttrs) {  
 scope.fun();//调用父作用域的方法，好似不能传参，未深究。  
 }  
 };  
 });  
</script>  
</body>  
  
</html>



## 4、link

运行levelone指令中的compile函数，ng就会递归遍历它的dom节点，然后在level-two与level-three上面重复这些操作。所以会依次打印连续三个compile。   
pre会在所有compile执行完后且在所有post之前执行。这样可以在执行post前执行一些其他代码，有些类似AOP。

post的执行顺序却是先levelthree最后levelone，即反向调用相关联的post-link函数。这么做的好处是，当我们运行levelone时，保证leveltwo与levelthree都已经执行过了，这样就会更安全。所以默认的link就是post。

<!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="myApp" ng-controller="myController">  
 <level-one>  
 <level-two>  
 <level-three> Hello </level-three>  
 </level-two>  
 </level-one>  
</div>  
<script>  
 //创建模块  
 var ***app*** = angular.module('myApp', []);  
 //创建控制器  
 ***app***.controller('myController', function($scope) {  
  
 });  
 //自定义指令  
 function *createDirective*(name){  
 return function(){  
 return {  
 restrict: 'E',  
 compile: function(tElem, tAttrs){  
 ***console***.log(name + ': compile => ' + tElem.html());  
 return {  
 pre: function(scope, iElem, iAttrs){  
 ***console***.log(name + ': pre link => ' + iElem.html());  
 },  
 post: function(scope, iElem, iAttrs){  
 ***console***.log(name + ': post link => ' + iElem.html());  
 }  
 }  
 }  
 }  
 }  
 }  
  
 ***app***.directive('levelOne', *createDirective*('levelOne'));  
 ***app***.directive('levelTwo', *createDirective*('levelTwo'));  
 ***app***.directive('levelThree', *createDirective*('levelThree'));  
</script>  
</body>  
  
</html>

### 注意打印结果：

levelOne: compile =>

<level-two>

<level-three>

Hello

</level-three>

</level-two>

levelTwo: compile =>

<level-three>

Hello

</level-three>

levelThree: compile =>

Hello

levelOne: pre link =>

<level-two>

<level-three>

Hello

</level-three>

</level-two>

levelTwo: pre link =>

<level-three>

Hello

</level-three>

levelThree: pre link =>

Hello

levelThree: post link =>

Hello

levelTwo: post link =>

<level-three>

Hello

</level-three>

levelOne: post link =>

<level-two>

<level-three>

Hello

</level-three>

</level-two>

# 7、页面跳转

## 1、ftl页面跳转 ctx=当做变量来存储

<#assign ctx=Request.request.getContextPath()>  
  
<div class="toolbar ad-flex exam">  
 <a class="first-btn first-btn-tab" href="${ctx}/chance#/new">新机会</a>  
 <a class="division"></a>  
 <a class="first-tab active" href="${ctx}/chance#/list">进行中  
 <span class="prompt">{{pageResult.totalElements}}</span>  
 </a>  
 <a class="first-tab" href="${ctx}/chance#/fail">已失败</a>  
 <a class="first-tab" href="${ctx}/chance#/file">已归档</a>  
</div>