# 引入

<script type="text/javascript" src="../js/angular.min.js"></script>

<script src="http://apps.bdimg.com/libs/angular-route/1.3.13/angular-route.js"></script>

<script src="http://apps.bdimg.com/libs/angular.js/1.4.6/angular-animate.min.js"></script>

# 基本指令

## ng-init

|  |
| --- |
| <div ng-app=*""* ng-init=*"name='hello';area='北京'"*>  <p>名称：<input type=*"text"* ></p>  <h1>{{name+area}}</h1>  </div |

## ng-app,ng-model,ng-bind

* ng-app 指令告诉 AngularJS，<div> 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。

|  |
| --- |
| <div ng-app=*""* ng-init=*"name='hello';area='北京'"*>  <p>名称：<input type=*"text"* ng-model=*"param"*></p>  <h1>{{name+area+param}}</h1>  <h2 ng-bind=*"name+area+param"*></h2>  </div> |

## ng-repeat

|  |
| --- |
| <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> |

## ng-options

|  |
| --- |
| $scope.sites = [  {site : "Google", url : "http://www.google.com"},  {site : "Runoob", url : "http://www.runoob.com"},  {site : "Taobao", url : "http://www.taobao.com"}  ];  $scope.cars = {  car01 : {brand : "Ford", model : "Mustang", color : "red"},  car02 : {brand : "Fiat", model : "500", color : "white"},  car03 : {brand : "Volvo", model : "XC90", color : "black"}  };   * 方式一：   <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>   * 方式二：   //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>   * 方式三：   //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  <select ng-model="selectedCar" ng-options="y.brand for (x, y) in sites">  </select> |

## ng-disabled, ng-show , ng-hide

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

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

* ng-show

ng-show 指令隐藏或显示一个 HTML 元素。

|  |
| --- |
| <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> |

* ng-hide

ng-hide 指令用于隐藏或显示 HTML 元素。

|  |
| --- |
| <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> |

## ng-include

|  |
| --- |
| <body>  <div class="container">  <div ng-include="'myUsers\_List.htm'"></div>  <div ng-include="'myUsers\_Form.htm'"></div>  </div>  </body> |

## 创建自定义的指令

### 定义

|  |
| --- |
| <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> |

### 使用

* 元素名<runoob-directive></runoob-directive>
* 属性<div runoob-directive></div>
* 类名<div class="runoob-directive"></div>
* 注释<!-- 指令: runoob-directive -->

### 限制使用

//通过添加 restrict 属性,并设置只值为 "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>"  };  }); |

* restrict 值可以是以下几种:

E 只限元素名使用

A 只限属性使用

C 只限类名使用

M 只限注释使用

restrict 默认值为 EA, 即可以通过元素名和属性名来调用指令。

# 表达式

## 数字

|  |
| --- |
| <div ng-app="" ng-init="quantity=1;cost=5">  <p>总价： {{ quantity \* cost }}</p>  </div> |

## 字符串

|  |
| --- |
| <div ng-app="" ng-init="firstName='John';lastName='Doe'">  <p>姓名： {{ firstName + " " + lastName }}</p>  </div> |

## 对象

|  |
| --- |
| <div ng-app="" ng-init="firstName='John';lastName='Doe'">  <p>姓名： {{ firstName + " " + lastName }}</p>  </div> |

## 数组

|  |
| --- |
| <div ng-app="" ng-init="firstName='John';lastName='Doe'">  <p>姓名： {{ firstName + " " + lastName }}</p>  </div> |

# 服务(Service)

## 内建服务

|  |  |
| --- | --- |
|  |  |
| $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。 |
| $timeout 服务 | AngularJS $timeout 服务对应了 JS window.setTimeout 函数。 |
| $interval 服务 | AngularJS $interval 服务对应了 JS window.setInterval 函数。 |
| $parse服务 | <div ng-app=*"myApp"* ng-controller=*"myCtl"*>  <input ng-model=*"expr"* type=*"text"* placeholder=*"Enter an expression"* />  <h2>{{ parseValue }}</h2>  </div>  <script type=*"text/javascript"* src=*"../js/angular.min.js"*></script>  <script type=*"text/javascript"*>  **var** app = angular.module('myApp',[]);  app.controller('myCtl',**function**($http,$scope,$parse){  $scope.$watch('expr', **function**(newVal, oldVal) {  **if** (newVal !== oldVal) {  // 用该表达式设置parseFun  **var** parseFun = $parse(newVal);  // 获取经过解析后表达式的值  $scope.parseValue = parseFun();  }  });  })  </script> |
|  |  |
|  |  |

## 自定义服务

### 创建

创建名为hexafy 的访问:

app.service('hexafy', function() {

this.myFunc = function (x) {

return x.toString(16);

}

});

### 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);

});

### 过滤器中使用自定义服务

* 加到过滤器中

app.filter('myFormat',['hexify', function(hexify) {

return function(x) {

return hexify.myFunc(x);

};

}]);

* 过滤器中使用

<ul>

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

</ul>

# 路由

## demo

|  |
| --- |
| smartApp.config(['$routeProvider', '$provide', **function**($routeProvider, $provide) {  $routeProvider  .when('/', {  redirectTo: '/dashboard'  })  /\* We are loading our views dynamically by passing arguments to the location url \*/  // A bug in smartwidget with angular (routes not reloading).  // We need to reload these pages everytime so widget would work  // The trick is to add "/" at the end of the view.  // http://stackoverflow.com/a/17588833  .when('/:page', { // we can enable ngAnimate and implement the fix here, but it's a bit laggy  templateUrl: **function**($routeParams) {  **return** 'views/'+ $routeParams.page +'.html';  },  controller: 'PageViewController'  })  .when('/:page/:child\*', {  templateUrl: **function**($routeParams) {  **return** 'views/'+ $routeParams.page + '/' + $routeParams.child + '.html';  },  controller: 'PageViewController'  })  .otherwise({  redirectTo: '/dashboard'  });  // with this, you can use $log('Message') same as $log.info('Message');  $provide.decorator('$log', ['$delegate',  **function**($delegate) {  // create a new function to be returned below as the $log service (instead of the $delegate)  **function** logger() {  // if $log fn is called directly, default to "info" message  logger.info.apply(logger, arguments);  }  // add all the $log props into our new logger fn  angular.extend(logger, $delegate);  **return** logger;  }]);  }]); |

## $routeProvider参数说明

$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。

# 过滤器

|  |  |
| --- | --- |
| 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。 |

## Date

* 本地化的日期格式

|  |
| --- |
| {{ today | date:'medium' }} <!-- Aug 09, 2013 12:09:02 PM -->  {{ today | date:'short' }} <!-- 8/9/1312:09PM -->  {{ today | date:'fullDate' }} <!-- Thursday, August 09, 2013 -->  {{ today | date:'longDate' }} <!-- August 09, 2013 -->  {{ today | date:'mediumDate' }}<!-- Aug 09, 2013 -->  {{ today | date:'shortDate' }} <!-- 8/9/13 -->  {{ today | date:'mediumTime' }}<!-- 12:09:02 PM -->  {{ today | date:'shortTime' }} <!-- 12:09 PM --> |

* 年份格式化

|  |
| --- |
| 四位年份：{{ today | date:'yyyy' }} <!-- 2013 -->  两位年份：{{ today | date:'yy' }} <!-- 13 -->  一位年份：{{ today | date:'y' }} <!-- 2013 --> |

* 月份格式化

|  |
| --- |
| 英文月份：{{ today | date:'MMMM' }} <!-- August -->  英文月份简写：{{ today | date:'MMM' }} <!-- Aug -->  数字月份：{{ today |date:'MM' }} <!-- 08 -->  一年中的第几个月份：{{ today |date:'M' }} <!-- 8 --> |

* 日期格式化

|  |
| --- |
| 数字日期：{{ today|date:'dd' }} <!-- 09 -->  一个月中的第几天：{{ today | date:'d' }} <!-- 9 -->  英文星期：{{ today | date:'EEEE' }} <!-- Thursday -->  英文星期简写：{{ today | date:'EEE' }} <!-- Thu --> |

* 小时格式化

|  |
| --- |
| 24小时制数字小时：{{today|date:'HH'}} <!--00-->  一天中的第几个小时：{{today|date:'H'}} <!--0-->  12小时制数字小时：{{today|date:'hh'}} <!--12-->  上午或下午的第几个小时：{{today|date:'h'}} <!--12--> |

* 分钟格式化

|  |
| --- |
| 数字分钟数：{{ today | date:'mm' }} <!-- 09 -->  一个小时中的第几分钟：{{ today | date:'m' }} <!-- 9 --> |

* 秒数格式化

|  |
| --- |
| 数字秒数：{{ today | date:'ss' }} <!-- 02 -->  一分钟内的第几秒：{{ today | date:'s' }} <!-- 2 -->  毫秒数：{{ today | date:'.sss' }} <!-- .995 --> |

* 字符格式化

|  |
| --- |
| 上下午标识：{{ today | date:'a' }} <!-- AM -->  四位时区标识：{{ today | date:'Z' }} <!--- 0700 --> |

* 示例：

{{ today | date:'MMMd, y' }} <!-- Aug9, 2013 -->

{{ today | date:'EEEE, d, M' }} <!-- Thursday, 9, 8-->

{{ today | date:'hh:mm:ss.sss' }} <!-- 12:09:02.995 -->

## filter

* 过滤条件为字符串

{{ ['Ari','Lerner','Likes','To','Eat','Pizza'] | filter:'e' }}

<!-- ["Lerner","Likes","Eat"] -->

* 过滤条件为对象

|  |
| --- |
| {{ [{  'name': 'Ari',  'City': 'San Francisco',  'favorite food': 'Pizza'  },{  'name': 'Nate',  'City': 'San Francisco',  'favorite food': 'indian food'  }] | filter:{'favorite food': 'Pizza'} }}  <!-- [{"name":"Ari","City":"SanFrancisco","favoritefood":"Pizza"}] --> |

* 过滤条件为函数

|  |
| --- |
| {{ ['Ari','likes','to','travel'] | filter:isCapitalized }}  <!-- ["Ari"] -->  isCapitalized函数的功能是根据首字母是否为大写返回true或false，具体如下所示：  $scope.isCapitalized = function(str) {  return str[0] == str[0].toUpperCase();  }; |

## Json

|  |
| --- |
| json过滤器可以将一个JSON或JavaScript对象转换成字符串。这种转换对调试非常有帮助：  {{ {'name': 'Ari', 'City': 'SanFrancisco'} | json }}  <!--  {  "name": "Ari",  "City": "San Francisco"  }  --> |

## limitTo

|  |
| --- |
| * 前三个字符   {{ San Francisco is very cloudy | limitTo:3 }}  <!-- San -->   * 最后的六个字符   {{ San Francisco is very cloudy | limitTo:-6 }}  <!-- cloudy -->   * 数组的第一个元素   {{ ['a','b','c','d','e','f'] | limitTo:1 }}  <!-- ["a"] --> |

# AngularJS全局API

## 转换

|  |  |
| --- | --- |
| API | 描述 |
| angular.lowercase() | 转换字符串为小写 |
| angular.uppercase() | 转换字符串为大写 |
| angular.forEach() | 对象或数组的迭代函数 |
| angular.copy() | 数组或对象深度拷贝 |

## 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 |

## 事件

ng-click

ng-dbl-click

ng-mousedown

ng-mouseenter

ng-mouseleave

ng-mousemove

ng-keydown

ng-keyup

ng-keypress

ng-change