# angularjs学习

## 一、简介

核心思想

1. 依赖注入 (参数注入$scope在controller中注入即可)
2. 模块化
3. 双向绑定 （{{ }} 的使用）
4. 语义化标签

## 二、如何引入

AngularJS 2.0 版本还没发布，但官方已经为其准备了全新版本的网站 <https://angular.io/>

而以前 AngularJS 1.x 的网站为 <https://angularjs.org/> 。

从新版网站中可下载 2.0 Alpha 版本：<https://angular.io/download/>

同时包含 2.0 的文档：<https://angular.io/docs/js/latest/>

在<script>标签里面引用文件即可

## 三、angularJs web应用

### 1.简单的入门例子

<!DOCTYPE html>  
<html lang="en" ng-app>  
<head>  
 <meta charset="UTF-8">  
 <title>angularjs</title>  
 <script src="http://code.angularjs.org/angular-1.0.1.min.js"></script>  
 <script type="text/javascript" src="src/lib/angular.min.js"></script>  
</head>  
<body>  
 <div class="main">  
 hello {{'world'}}  
 </div>  
</body>  
</html>

用浏览器运行代码即可

### 2.指令的使用

#### 1）ng-app

声明AngularJs管理边界，标记ng-app告诉AngularJS处理整个HTML页并引导应用。

ng-app指令标记了AngularJS脚本的作用域，在<html>中添加ng-app属性即说明整个<html>都是AngularJS脚本作用域。开发者也可以在局部使用ng-app指令，如<div ng-app>，则AngularJS脚本仅在该<div>中运行。

#### 2）双向绑定

{{ }} 和 ng-model指令配合使用

<!DOCTYPE html>  
<html lang="en" ng-app>  
<head>  
 <meta charset="UTF-8">  
 <title>angularjs双向绑定</title>  
 <script type="text/javascript" src="src/lib/angular.min.js"></script>  
</head>  
<body>  
 <div>  
 your name:<input ng-model="yourname" placeholder="请输入名字"/>  
 <hr>  
 Hello {{yourname || 'World'}}!  
 </div>  
</body>  
</html>

ng-init初始化样式

ng-init=’hello=kitty’

#### 3）ng-bind

绑定值

<div class="main" ng-controller="MsgCtrl">  
 <!--view层-->  
 <input ng-model="msg" value="{{msg}}"/>  
 <h2 ng-bind="msg"></h2>  
 hello {{msg}}  
</div>

#### 4）作用域 ($scope)

//console.log(angular); angular对象  
**var** app = angular.module('web',[]);  
console.log(app);  
  
angular.module('web',[])  
 .controller('MsgCtrl', **function** ($scope) {  
 //model层  
 $scope.msg = "angular";  
  
 }).controller("demoController", **function** ($http, $scope) {  
 $scope.getAjaxUser = **function**(){  
 //ajax请求  
 // $http.get({url:"../xxx.action"}).success(function(data){  
 // $scope.user= data;  
 // });  
 $scope.user = {"name":"从JOSN中获取的名称","age":22};  
 }  
 });

angular中$scope是一种服务，提供作用域的功能。

**注意**：在angular中内置的服务都是$开头的，就像angular中指令移以ng-开头

#### 5）事件指令

angular中点击事件方法需要定义在作用域内

事件指令如下：

--- ng-click

更改数据的方式：

$scope和this调用

#### 6)显示隐藏

ng-show

ng-hide

#### 7)插值表达式

ng-bind 指令可以使用表达式

#### 8）过滤器

过滤器是为了视图准备的，解决页面臃肿的问题和表达式复用问题，

作用就是接收一个输入，通过某个规则进行处理，然后返回处理后的结果

ng内置了一些过滤器，它们是：

currency(货币)、

date(日期)、

filter(子串匹配)、

json(格式化json对象)、

limitTo(限制个数)、

lowercase(小写)、

uppercase(大写)、

number(数字)、

orderBy(排序)。

总共九种。除此之外还可以自定义过滤器，这个就强大了，可以满足任何要求的数据处理。

##### 1. 在模板中使用filter(视图层)

我们可以直接在{{}}中使用filter，跟在表达式后面用 | 分割，语法如下：

**{{ expression | filter }}**

也可以多个filter连用，上一个filter的输出将作为下一个filter的输入（怪不得这货长的跟管道一个样。。）

**{{ expression | filter1 | filter2 | ... }}**

filter可以接收参数，参数用 : 进行分割，如下：

{{ **expression** | **filter**:argument1:argument2:... }}

除了对{{}}中的数据进行格式化，我们还可以在指令中使用filter，例如先对数组array进行过滤处理，然后再循环输出：

<span ng-**repeat**="a in array | filter ">

##### 2. 在controller和service中使用filter（控制器和业务逻辑层）

我们的js代码中也可以使用过滤器，方式就是我们熟悉的依赖注入，例如我要在controller中使用currency过滤器，只需将它注入到该controller中即可，代码如下：

app.controller('testC',**function**($scope,currencyFilter){

$scope.num = currencyFilter(123534);

}

在模板中使用{{num}}就可以直接输出 $123,534.00了！在服务中使用filter也是同样的道理。

此时你可能会有疑惑，如果我要在controller中使用多个filter，难道要一个一个注入吗，这岂不太费劲了？小兄弟莫着急~ng提供了一个$filter服务可以来调用所需的filter，你只需注入一个$filter就够了，使用方法如下：

app.controller('testC',**function**($scope,$filter){

$scope.num = $filter('currency')(123534);  
　　$scope.date = $filter('date')(**new** Date());

}

可以达到同样的效果。好处是你可以方便使用不同的filter了。

##### ng的内置过滤器

ng内置了九种过滤器，使用方法都非常简单，看文档即懂。不过为了以后不去翻它的文档，我在这里还是做一个详细的记录。

###### 1. currency (货币处理)

使用currency可以将数字格式化为货币，默认是美元符号，你可以自己传入所需的符号，例如我传入人民币：

{{num | currency : '￥'}}

###### 2. date (日期格式化)

原生的js对日期的格式化能力有限，ng提供的date过滤器基本可以满足一般的格式化要求。用法如下：

{{date | date : 'yyyy-MM-dd hh:mm:ss EEEE'}}

参数用来指定所要的格式，y M d h m s E 分别表示 年 月 日 时 分 秒 星期，你可以自由组合它们。也可以使用不同的个数来限制格式化的位数。另外参数也可以使用特定的描述性字符串，例如“shortTime”将会把时间格式为 12:05 pm这样的。ng提供了八种描述性的字符串，个人觉得这些有点多余，我完全可以根据自己的意愿组合出想要的格式，不愿意去记这么多单词~

###### 3. filter(匹配子串)

这个名叫filter的filter（不得不说这名字起的，真让人容易混淆——！）用来处理一个数组，然后可以过滤出含有某个子串的元素，作为一个子数组来返回。可以是字符串数组，也可以是对象数组。如果是对象数组，可以匹配属性的值。它接收一个参数，用来定义子串的匹配规则。下面举个例子说明一下参数的用法，我用现在特别火的几个孩子定义了一个数组：

$scope.childrenArray = [

{name:'kimi',age:3},

{name:'cindy',age:4},

{name:'anglar',age:4},

{name:'shitou',age:6},

{name:'tiantian',age:5}

];

$scope.func = **function**(e){**return** e.age>4;}

{{ childrenArray | filter : 'a' }} //匹配属性值中含有a的

{{ childrenArray | filter : 4 }} //匹配属性值中含有4的

{{ childrenArray | filter : {name : 'i'} }} //参数是对象，匹配name属性中含有i的

{{childrenArray | filter : func }} //参数是函数，指定返回age>4的

###### 4. json(格式化json对象)

json过滤器可以把一个js对象格式化为json字符串，没有参数。这东西有什么用呢，我一般也不会在页面上输出一个json串啊，官网说它可以用来进行调试，嗯，是个不错的选择。或者，也可以用在js中使用，作用就和我们熟悉的JSON.stringify()一样。用法超级简单：

{{ jsonTest | json}}

###### 5. limitTo(限制数组长度或字符串长度)

limitTo过滤器用来截取数组或字符串，接收一个参数用来指定截取的长度。个人觉得这个filter有点鸡肋，首先只能从数组或字符串的开头进行截取，其次，js原生的函数就可以代替它了，看看怎么用吧：

{{ childrenArray | limitTo : 2 }} //将会显示数组中的前两项

###### 6. lowercase(小写)

把数据转化为全部小写。太简单了，不多解释。同样是很鸡肋的一个filter，没有参数，只能把整个字符串变为小写，不能指定字母。怎么用我都懒得写了。

###### 7. uppercase(大写)

###### 8. number(格式化数字)

number过滤器可以为一个数字加上千位分割，像这样，123,456,789。同时接收一个参数，可以指定小float类型保留几位小数：

{{ num | number : 2 }}

###### 9. orderBy(排序)

orderBy过滤器可以将一个数组中的元素进行排序，接收一个参数来指定排序规则，参数可以是一个字符串，表示以该属性名称进行排序。可以是一个函数，定义排序属性。还可以是一个数组，表示依次按数组中的属性值进行排序（若按第一项比较的值相等，再按第二项比较），还是拿上面的孩子数组举例：

<div>{{ childrenArray | orderBy : 'age' }}</div> //按age属性值进行排序

<div>{{ childrenArray | orderBy : orderFunc }}</div> //按照函数的返回值进行排序

<div>{{ childrenArray | orderBy : ['age','name'] }}</div> //如果age相同，按照name进行排序

内置的过滤器介绍完了，写的我都快睡着了。。。正如你所看到的，ng内置的过滤器也并不是万能的，事实上好多都比较鸡肋。更个性化的需求就需要我们来定义自己的过滤器了，下面来看看如何自定义过滤器。

###### 自定义过滤器

filter的自定义方式也很简单，使用module的filter方法，返回一个函数，该函数接收输入值，并返回处理后的结果。话不多说，我们来写一个看看。比如我需要一个过滤器，它可以返回一个数组中下标为奇数的元素，代码如下：

app.filter('odditems',**function**(){

**return** **function**(inputArray){

**var** **array** = [];

**for**(**var** i=0;i<inputArray.length;i++){

**if**(i%2!==0){

**array**.push(inputArray[i]);

}

}

**return** **array**;

}

});

格式就是这样，你的处理逻辑就写在内部的那个闭包函数中。你也可以让自己的过滤器接收参数，参数就定义在return的那个函数中，作为第二个参数，或者更多个参数也可以。

过滤器的基本知识就这些了。还是那句话，更多需要学习的还需项目的真正考验。那么，在项目来临之前，先打好基础吧~

（PS：忍不住要抱怨了，使用博客园的编辑器编辑的例子不知什么原因抽风不能动了，以前还是好好的。上一篇我试着使用了jsfiddle提供的内嵌代码，但访问速度又慢的跟蜗牛似的。唉，哪里有个好用的工具可以让我在博客中内嵌示例啊~）