# Angular

Angular2：https://www.angular.cn/guide/quickstart

* 官网： <http://angularjs.org> 官方文档 ：<https://code.angularjs.org/1.6.4/docs/api>
* 中文文档 ：<http://www.angularjs.net.cn> http://www.runoob.com/angularjs/angularjs-tutorial.html
* ng // angular的简称
* **ng-app**  // 告诉angular你可以去执行了
* **ng-model=”变量”**  // 获取文本框内容
* **ng-init=”val = ‘ 我是变量val的初始值 ’ ”** // 设置变量的初始值
* **{{ }}**  // 差值表达式

作用 ：

1. 解析angular 当中的变量 // {{ 变量名 }}
2. 输出字符串 // {{ ‘ 我是字符串 ’ }}
3. 计算 // {{ ( 变量-0 ) + 1 }}
4. 三元运算 // {{ 1+1==2 ? ‘ 正确 ’ : ‘ 错误 ’ }}
5. 执行函数 // {{ fn() }}

* angular 模块开发

案例说明：

<body **ng-app=”模块名”** > // 告诉angular 用我自己创建的模块管理页面

<div class=”siderBar” **ng-controller=”控制器的名字A”**> // 告诉angular 我创建的控制器管理页面中的哪个区域

**{{ val }}**  // 输出模块里的定义的val变量的值

**< button ng-click=”fun()” ></button>**  // 绑定模块里的方法

</div>

<div class=”main” **ng-controller=”控制器的名字B”**> // 告诉angular 我创建的控制器管理页面中的哪个区域

</div>

</body>

<script src=”./angular.js”></script> // 引包

<script>

// 创建模块

// 参数1：模块的名字 字符串类型

// 参数2：当前模块依赖的其他的模块的名字 数组类型

**var myapp = angular. module( ‘ 模块名 ’ , [ ] )**

**myapp.controller( ‘ 控制器的名字A ’,[‘$scope’,function ( $scope ) {** // 控制器创建法

**$scope.val = ‘ 哈哈哈 ’**

**$scope.fun = function(){**

alert( 方法 )

**}**

**} ])**

**myapp.controller( ‘ 控制器的名字B ’,[‘$scope’,function ( $scope ) {** // 控制器创建法

**} ])**

</script>

* 创建模块与获取模块
  + **var aaa = angular.module(‘ 模块名 ’,[ ])**  // 创建模块
  + **angular.module(‘ 模块名 ’)**  // 获取模块

aaa == angular.module(‘ 模块名 ’) // trrue

* **Angular中的路由** 
  + 作用：实现单页面dom

案例说明： 此时 有三个文件 angular.html index.html list.html

angular.html ：

<a href=”**#!/index**”></a> // 两个a标签元素 href的值锚点链接的是两个页面

<a href=”**#!/list**”></a> // 注意 描点#号后面一定要加是!

<div **ng-view**></div> /**/ ng-view**  // 是吧外部的文件内容放到这里

<script src=”./angular.js”> // 引包

<script src=”./angular-route.js”> // 引路由包

<script>

**var myApp = angular.module( ‘ myapp ’, [ ‘ ngRoute ’ ] )** //创建angular对象

myApp.**config**( [ ‘ $routeProvider ’ , function( $routeProvider ){ // **config()**  配置方法

$routeProvider.when( ‘ /index ’ ,{ // **when()** 链接方法 /index a标签的描点值

**templateUrl :** ‘ ./index.html ’ // 链接到外部文件 index.html

}).when( ‘ /list ’ , {

**templateUr**l : ‘ ./list.html ’ // 链接到外部文件 list.html

} )

} ])

</script>

* **第三方强大路由** [**https://github.com/angular-ui/ui-router**](https://github.com/angular-ui/ui-router)
  + 页面里：

< a **ui-sref=”index”**> 首页 < /a > // 新的路由命令 这个命令可以自动生成href=”#!/index” 也可以直接使用href

< div **ui-view** >< /div > // 新命令 把文件放在这个命令的位置

* + 语法：

模块对象.config( [ ‘ **$stateProvider** ’ , function( **$stateProvider** ){ // **$stateProvider** 新的路由对象

**$stateProvider.state(**{ // 变成了对象形式

name : ‘ index ’ , // 必填属性 对应的是 页面命令ui-sref=”index”

url : ‘ /index ’ , // 页面中的锚点名

templateUrl : ‘ ./index.html ’, // 实际的外部文件路径

controller : ‘ indexCtrl ’ // 控制器的名字

})

}] )

* 页面中有两个或多个**ui-view**
  + - 案例：< a ui-sref=”index” >< /a > < div ui-view=”view-a” >< /div > < div ui-view=”view-a” >< /div >

angular.module ( ‘ myapp ’ , [ ‘ ui.router ’ ] ) . config ( [ ‘**$stateProvider** ’ , function( **$stateProvider** ){

**$stateProvider . state( {**

name : ‘ index ’ , // 必填属性 对应的是 页面命令ui-sref=”index”

url : ‘ /index ’ , // 页面中的锚点名

**views :** { // 放一个对象

**view-a :** { // 页面中 ui-view对应的名称

template : ‘ <div>我是view-a 中的内容<div> ’ // 写上对应的文件或路径

} ,

**view-b :** { // 页面中 ui-view对应的名称

template : ‘ <div>我是view-b 中的内容<div> ’ // 写上对应的文件或路径

}

}

controller : ‘ indexCtrl ’ // 控制器的名字

**} )**

.otherwise ( ‘ /index ’ ) **//** 当访问一个不存在的路径的时候 默认跳转到首页的方法 （不完善）

} ] )

* 路由的嵌套
  + - 案例 （首页有导航 列表有导航 列表详情没有导航）
    - **$urlRouterProvider**  // 当访问一个不存在的路径的时候 默认跳转到首页的对象

Html : < div ui-view >< /div >

angular.module ( ‘ myapp ’ , [ ‘ ui.router ’ ] )

. config ( [ ‘**$stateProvider** ’ , ‘ $urlRouterProvider ’ , function( **$stateProvider** , $urlRouterProvider){

**$stateProvider . state( {**

name : ‘ tab ’ , // 必填属性 对应的是 页面命令ui-sref=”index”

url : ‘ /tab ’ , // 页面中的锚点名

**abstract : true ,**  // 作用是不能单独使用此页面

// 为了导航在首页 列表页显示 所以要关联起来 后边的 **< div ui-view >< /div >** 是用来放置首页和列表页内容的 因为一个**ui-view** 只能放一个页面的

template : ‘ <a **ui-sref=”tab.index”**>首页</a><a **ui-sref=”tab.list”**>列表页</a> **< div ui-view >< /div >**’

**} ) . state( {**

name : ‘ **tab.index** ’ , // tab与首页关联

url : ‘ /index ’ ,

template : ‘ <div>我是首页<div> ’

**} ) . state( {**

name : ‘ **tab.list** ’ , // tab与列表页关联

url : ‘ /list ’ ,

template : ‘ <div>我是列表页<div> ’

**} ) . state( {**

name : **‘ listt** ’ , // tab与列表详情无关联

url : ‘ /listt ’ ,

template : ‘ <div>列表详情页<div> ’

**} )**

**$urlRouterProvider.otherwise( ‘ tab/index ’ )**

} ] )

* **控制器与路由的关联 以及路由传参 （ 详情可查看笔记跟随案例 ）**
  + **$routeParams** // 接受地址栏里的数据 主要记住这个参数
  + 案例说明： 此时有三个文件 index.html list.html details.html

angular.html:

< a href=”**#!/**index” >我是首页< /a > // 利用锚点跳转页面 angular中锚点的固定写法 #!/

< a href=”**#!/**list” >我是列表< /a >

// 引包：

< script src=”./angular.js” >< /script >

< script src=”./angular-route.js” >< /script >

< script >

angular.module ( ‘ myapp ’ , [ ‘ ngRoute ’ ] ) // 模块对象

. config ([ ‘ $routeProvider ’ , function( $routeProvider ){ // 路由

$routeProvider . when ( ‘ /index ’ , {

templateUrl : ‘ ./index.html ’ ,

**controller : ‘ indexCtrl ’**  // 控制器的名字

} ) . when ( ‘ /list ' , {

templateUrl : ‘ ./list.html ’ ,

**controller : ‘ listCtrl ’** // 控制器的名字

}) . when ( ‘ /details**/:id** ' , { // 这里的**/:id**是传递给下一个页面的数据参数 传递 多个可以写成 **:/id/:name**

templateUrl : ‘ ./details.html ’ ,

**controller : ‘ detailsCtrl ’** // 控制器的名字

})

} ] )

.controller ( ‘ indexCtrl ’ , [ ‘ $scope ’ , function( $scope ) {

// 这里可以写index.html文件 的逻辑代码

} ] )

.controller ( ‘ listCtrl ’ , [ ‘ $scope ’ , function( $scope ) {

// 这里可以写list.html文件 的逻辑代码

$scope.data = [ // 传递参数用

{ id : 1 , name : ‘ 李艳东 ’ , age : 18 } ,

{ id : 2 , name : ‘ 李艳影 ’ , age : 20 } ,

{ id : 3 , name : ‘ 李艳娟 ’ , age : 22 } ,

]

} ] )

.controller ( ‘ detailsCtrl ’ , [ ‘ $scope ’ , **‘ $routeParams ’** , function( $scope , **$routeParams** ) {

// 这里可以写details.html 的逻辑代码

console.log(**$routeParams** ); **// $routeParams** 是一个路由用来获取地址栏里的数据的一个对象

} ] )

< /script >

list.html 文件 里的内容 ：（让details文件 接受参数的方法）

< li ng-repeat = “ item in data ” > < a href=”**#!/details/{{ item.id }}**” > {{ item.name }} < /a > < /li >

* 常用指令

1. ng-style=”{ ‘width’:’100px’,’height’:’100px’... }” // 样式指令
2. ng-repeat=”( key , value) in 数组或对象” // 循环数组或对象
   1. **ng-repeat=”key in arrr track by $index”** // angular循环数组不允许出现重复项 这是解决办法
   2. {{ $index }} // 数组索引
   3. {{ $first }} // 第一项 返回布尔类型值 true
   4. {{ $last }} // 第一项 返回布尔类型值 true
   5. {{ $middle}} // 中间项项 返回布尔类型值 true （除了第一个 和最后一个 都是中间项）
   6. {{ $odd }} // 奇数项 返回布尔类型值 true
   7. {{ $event }} // 偶数项 返回布尔类型值 true

例： <li ng-repeat=”item in arr”>{{$index}} {{item} </li> // 输出索引 还有数组值

<li ng-repeat=”(key,value) in obj”>{{key}} {{value}}</li> // 输出对象键 还有值

<li ng-repeat=”value in arrObj”>{{value.name}}</li> // 输出数组里对象的name的值

<li ng-repeat=”value in arr” class=”{{ $first ? ‘first’,’’]]: ’’}}”>{{value}}</li> // 第一个li添加一个类名

1. **ng-class=”{ ‘ 类名 ’ 变量或表达式 }”** // 设置类名

例： <li ng-repeat=”item in arr” **ng-class**=**{** ‘ box ’:{{ $first }} , ’zhong’:{{ $middle }} , ‘ last ’ : {{ $last }} , ‘ suoyou ’ : 1==1 **}**></li>

1. **ng-bind=”变量”** // 在angular没有引进页面之前页面会把‘{{}}’给解析出来 这个命令就可以避免 但是只能绑定单个数据
2. **ng-bind-template=”{{ 变量1 }}{{ 变量2 }}...”** // 多数据绑定
3. **ng-cloak** // 样式绑定数据 需要在style样式表里写[ng-cloak]{display:none} 这种可以绑定多数据 也可以避免页面解析‘{{}}’
4. **ng-non-bindable** // 不解析数据
5. **ng-show=”变量”** // 控制元素显示 true 显示 false 隐藏

例：小tab栏 angulas 里面定义boot=”true”

<button ng-click=”boot=”true”>aaa</button> <div ng-show=”boot”>aaaaaaaaa<div>

<button ng-click=”boot=”false”>bbb</button> <div ng-show=”!boot”>bbbbbbbb<div>

1. **ng-hide=”变量”** // 控制元素隐藏 true 隐藏 false 显示
2. **ng-if=”变量”** // 控制元素删除添加 true 删除元素 false 添加元素
3. 多项table

案例：

<div **ng-init=”tab = 1” ng-switch=”table”**> // 定义变量

<button **ng-click=”tab=1”**>我对应第一个div</button> // 命令

<button **ng-click=”tab=2”**>我对应第二个div</button>

<button **ng-click=”tab=3”**>我对应第三个div</button>

<div **ng-switch-when=”1”**>我是第一个div</div> // 执行命令

<div **ng-switch-when=”2”**>我是第二个div</div>

<div **ng-switch-when=”3”**>我是第三个div</div>

</div>

1. **ng-options=” item.value as item.name for item in 数组 ”** // 用来设置下拉列表数据的命令
   * 1. 必须配合 **ng-model=”ball”**这个命令使用 ball是个变量 表示选中项 也可以设置默认值
     2. **item.name for**  // 固定语法 item.name 显示在option标签里的值
     3. **item.value as**  // 固定语法 item.value 显示在value 属性里的值
     4. 例 ：<select ng-model=”ball” ng-options=”item.value as item.name for item in balls”></select>

angular:

$scope.**ball** = ‘001’ // 设置select默认值

$scope.**balls** = [ // 案例数据

{name:’第一’,value:’001’},

{name:’第二’,value:’002’},

{name:’第三’,value:’003’}

]

1. **ng-click=”fn($event)”** // 单击事件 **$event //事件对象** 例：<div ng-click=”fn()”>aa</div>
2. **ng-mouseove=”fn($event)”** // 划上事件
3. **ng-dblclick=”fn($event)”** // 双击事件
4. **$scope.$apply()** // 告诉angular 数据有变化 就帮我更新一下 ajax 也会用到
   1. **解释：当我们通过定时器 或者 第三方库（发送ajax请求） 将angular中的数据做了更改的时候 那么我们就需要使用$scope $apply() 去通知angular我们的数据变了 你帮我更新一下**

例： <a href=”{{link}}”>

angular:

setTimeout (function(){

$scope.link = ‘ http://www.baidu.com ’

**$scope.$apply()** // 告诉angular 数据有变化 就更新一下

},2000)

1. **$timeout( function ( ) { } , 2000 )**  // angular 提供的单次定时器
2. **ng-href=”{{变量}}”** // 生成链接 防止无效链接
3. **ng-src=”{{变量}}”**  // 生成src属性 防止404 找不到图片文件

* 以面向对象的方式去创建控制器
  + 解释 ：angular会将控制器的处理函数当做 构造器 new obj 就是new出来的对象

使用的方法： ng-controller=” 控制器的名字 **as 构造函数的名字** ”

使用 ： <div ng-app=”myapp” ng-controller=”**demo as obj**”>

**{{ obj.name }}**  // 李艳东

{{ name }} // 李艳娟

</div>

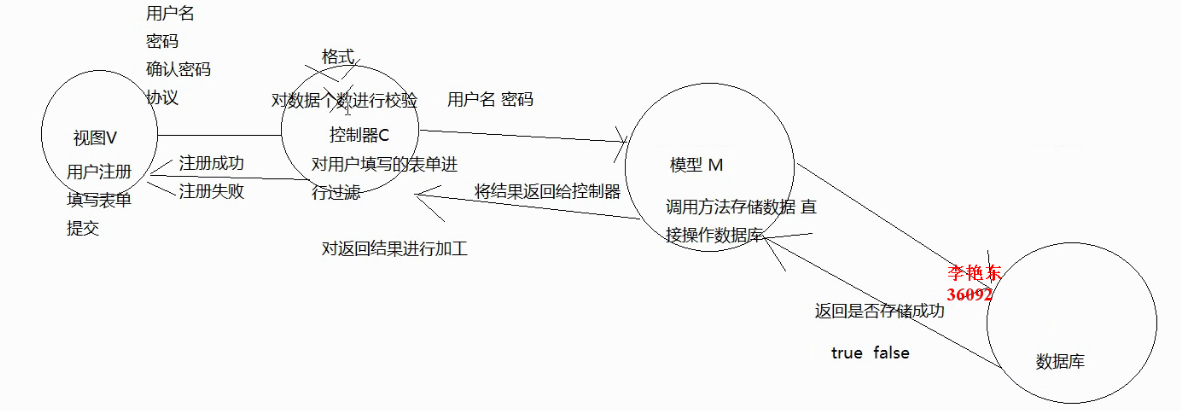
创建 ： angular.module( ‘ myapp ’ ,[]) . controller ( ‘ demo ’ , [ $scope , function ( ) {

**this.name = ‘ 李艳东 ’;**

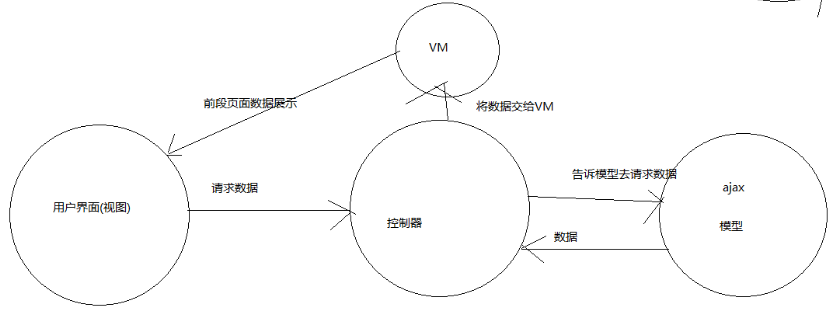
$scope.name = ‘ 李艳娟 ’

} ] )

* **MVC介绍**
  + MVC 思想愿与后端
  + M： model 模型 数据和数据相关的方法
  + V： view 视图 用户界面
  + C： controller 控制器 网站的业务逻辑都写在控制器里面
  + 总结 ： 对代码进行分类 便于管理
  + MVC思想并不完全适用于前端
  + MVC里面的控制器做的事情太多了 代码又变得冗多



* **MVVM思想**
  + 它并不是一个全新的思想 是对MVC思想的改进 升级
  + M : model 模型 数据和数据相关的方法 ajax请求
  + V : view 视图用户界面
  + VM : 它都是跟数据双向绑定有关系 $scope
  + Angular就是基于MVVM这样的的一个前端框



* Angular里提供一个简化的jq : **jqLite** jqLite提供一些和jq名字一样的方法
  + **angular.element( 原生js对象 )** // 将原生js对象转换成jqLite对象 这样的话我们就可以使用jqLite 里面的方法了

例： <div **id=”box”**></div>

angular.module( ‘ myapp ’ , [ ] ) . controller( ‘ one ’ , [ ‘ $scope ’ , function ( $scope ) {

**var box = document . getElementById( ‘ box ’ );**

**var jqLiteDiv = angular.element( box )**;

**jqLiteDiv.css( { background:’red ’ , fontSize:’ 30px ’ ...} );**

}] );

* 自定义属性

1. 指令名字 html : my-dir --> js : myDir
2. 模块对象.**directive(** ‘ 指令名称 ’ , [ function ( ) { // directive 创建自定义指令的固定写法

**return {** // 固定写法

**restrict : ‘ A ’** , // 限制指令的类型 默认是 ’ AE ’

**template: ’ <button ng-transclude></button> ’** , // 向指令内部添加自定义的html模板

**templateUrl : ‘ 文件的路径 ’ ，** // 向指令内部添加自定义的html模板 （内容多就要用外部文件）

templateUrl : ‘ 模板id ’ // 还可以定义script模板写模板id

**transclude : true ,**  // 保存指令所在标签内部的内容是否保留默认false不保留true 但要配合 ng-transclude命令保留到哪里

**replace:true,**  // template添加的模板 替换 掉带有my-dir指令的元素 默认false

**link : function ( scope , element , attributes ) {** // 写具体的指令要做的事情

// **scope指令** // 跟$scope 作用一样 但是范围不一样 scope生效范围 指令的template里面的自定义模板

// **element指令** // 所在的元素 jqLite对象 如果写了replace:true 替换了 就转到替换的元素上

//  **attributes指令** // 所在标签上的属性集合

**}**

**}**

} ] )

1. 指令分类：**restrict : ‘ AECM ’** 
   1. 属性指令 A attribute --> 例： < div my-dir>< /div >
   2. 元素指令 E element --> 例： < div my-dir>< /div >
   3. Class指令 C class --> 例： < div my-dir>< /div >
   4. 注释指令 M comment --> 例： <!-- directive:my-dir --> // 必须配合replace:true属性才能生效
   5. 作用
      1. DOM操作
      2. 模板复用
2. 综合写法

return function( scope , element , attribute ){

alert(‘11’)

}

5) 案例：回到顶部 查看笔记跟随案例

* Angular的数据请求 https://cdn.bootcss.com/angular.js/1.6.4/angular-sanitize.min.js

1. angular.module( ‘ myapp ’ , [ ] ) . controller( ‘ one ’ , [ **‘ $http ’** , function ( **$http** ) { // 创建angular对象 设定请求对象$http

**$http** ( {

**method** : ‘ post / get ’ , // 设置请求方式

**url :** ‘ 地址 ’ , // 设置请求地址

**params** : { name:’张三’ } // get请求提交的参数

**data** : { name : ‘ 张三 ’ } // post请求提交的参数

} )**.then**( function ( res ){ // then链接响应回来的数据

// 请求成功的程序

} , function ( res ) {

// 请求失败的程序

});

}] );

1. **$http.get( ‘ 地址 ’ , { params : { 传递的数据 } } ) . then ( function( res ){ 程序 } )**  // get 请求简写
2. **$http.post( ‘ 地址 ’ , { 传递的数据 } ) . then ( function( res ){ 程序 } )**  // post 请求简写
3. **ng-bind-html=”请求回来的数据”** // angular不会解析数据中的标签 需要用到这个命令 angular里没有这个命令 所以需 要使用一个angular插件 ngSanitize 文件里有

* 过滤器

介绍：概念：angular为我们提供的处理数据的格式的方式

分类：内置过滤器 自定义过滤器

使用语法： **{{ 数据模型 | 过滤器的名字 : 过滤器的参数 : 多个参数以冒号隔开 }}**

1. {{ num **| currency:’￥’** }} // 货币过滤器 num=988 // ￥988.00
2. {{ date **| date : ’ yyyy-MM-dd HH:mm:ss ’** }} // 时间过滤器 // 2017 - 06 - 05 03 : 50 : 20

{{ date **| date : ‘ yyyy年MM月dd日 HH时mm分ss秒 ’** }} // 2017年6月 5日 15时 30分59秒

1. <li ng-repeat=” item in arr **| filter : ’孙’** ” > {{ item.id }} {{ item.name }} </li> // 模糊匹配 // 会输出 : 2 孙悟空

<li ng-repeat=” item in arr **| filter : { name:’沙’ }** ” > {{ item.id }} {{ item.name }} </li> // 精确匹配 // 会输出 : 4 沙悟净

例 数据 $scope. arr = [

{ id:1 , name : ’ 唐三藏 ’ } ,

{ id:2 , name : ‘ 孙悟空 ’ } ,

{ id:3 , name : ‘ 猪八戒 ’ } ,

{ id:4 , name : ‘ 沙悟净 ’ }

];

1. {{ ‘ 陪我在成都的街头走一走 ’ **| limitTo:1** }} // 截取过滤器 // 陪 --> 从0开始 截取1个

{{ ‘ 陪我在成都的街头走一走 ’ **| limitTo:2:3** }} // 截取过滤器 // 成都 --> 从第3个开始 截取两个

<li ng-repeat=”item in arr **| limitTo:2**”>{{ item.name }}</li> // 唐三藏 孙悟空 --> 从第0开始 截取两个显示

例 数据 $scope. arr = [

{ id:1 , name : ’ 唐三藏 ’ } ,

{ id:2 , name : ‘ 孙悟空 ’ } ,

{ id:3 , name : ‘ 猪八戒 ’ } ,

{ id:4 , name : ‘ 沙悟净 ’ }

];

1. <li ng-repeat=”item in arr **| orderBy : ’age’**”>{{ item.name }} {{ item.age }}</li> // 排序默认小到大
2. <li ng-repeat=”item in arr **| orderBy : ’-age’**”>{{ item.name }} {{ item.age }}</li> // 加个‘-’号排序大到小

例 数据 $scope. arr = [

{ id:1 , name : ’ 唐三藏 ’ , **age** : 300} ,

{ id:2 , name : ‘ 孙悟空 ’ , **age** : 800 } ,

{ id:3 , name : ‘ 猪八戒 ’ , **age** : 500 } ,

{ id:4 , name : ‘ 沙悟净 ’ , **age** : 100 }

];

1. < pre > {{ arr | json }} < / pre > // 把数据清晰的展示在页面中

< pre > {{ arr | json : 5 }} < / pre > // 把数据清晰的展示在页面中 加上数字代表 数据前面的空格有多少

例 数据 $scope. arr = [

{ id:1 , name : ’ 唐三藏 ’ , **age** : 300} ,

{ id:2 , name : ‘ 孙悟空 ’ , **age** : 800 } ,

{ id:3 , name : ‘ 猪八戒 ’ , **age** : 500 } ,

{ id:4 , name : ‘ 沙悟净 ’ , **age** : 100 }

];

1. {{ ‘ 123456789 ’ **| number** }} // 数字过滤器 良好的显示 --> 123,456,789

{{ ‘ 123456789 ’ **| number : 3**}} // 后面加数字代表小数点后面的个数 --> 123.456,789.000

1. {{ ‘ abc ’ **| uppercase** }} // 转换为大写字母 --> ABC
2. {{ ‘ ABC ’ **| lowercase** }} // 转换为小写字母 --> abc

* 自定义过滤器

语法：

**模块对象 . filter ( ‘ 过滤器名字 ’ , [ function ( ) {**

**return funcction ( value ) {**

**// value 处理的数据**

**// 将处理完的数据 return**

**}**

**} ] )**

案例： 把手机号码中间四位数变成 \*\*\*\* 单词首字母大写

使用： {{ shouji **| phon** }} // 158 \*\*\*\* 0147

{{ an **| toupp** }} // Angular

angular . moudle ( ‘ myapp ’ , [ ] ) . controller ( ‘ one ’ , [ ‘ $scope ’ , function ( $scope ) {

$scope . shouji = ‘ 15810100147 ’;

$scope . an = angular;

} ] ) **. filter ( ‘ phon ’ , [ function ( ) { // 手机号码中间四位变成 \*\*\*\***

**return function ( value ) {**

**var front = value . substr( 0 , 3 ) ;** // 截取电话号码的前三位数字

**var back = value . substr ( 7 ) ;** // 截取电话号码的后四位数字

**return front + ‘ \*\*\*\* ’ + back ;**

**}**

**} ] ).filter( ‘ toupp ’ , [ function ( ) { // 单词首字母大写 链式编程**

**return function ( value ) {**

**var first = value.substr( 0 , 1 ) . toUpperCase( );**  // 截取第一位字母 并且改成大写

**var last = value.substr ( 1 );** // 截取除了第一位字母剩下的所有字母

**return first + last;**

**}**

**} ] );**

* **$scope.$watch( ‘ 要监控的数据 ’ , function (参数1 , 参数2 ){ } )** // 监控数据 页面一上来 默认执行一次
  + 参数1 // 是数据的当前值
  + 参数2 // 是数据改变之前上一次的值

案例： 做表单验证

html : < label > < input type=”text” ng-model=”val” > {{ yanzheng }} < /label >

js : angular.module( ‘ myapp ’ , [ ] ) . controller ( ‘ one ’ , [ ‘ $scope ’ , function ( $scope ) {

$scope.**val** = ‘ 我是默认值 ’;

$scope.yanzheng = ‘ ’;

**$scope.$watch(** ‘ **val** ’ , function ( newValue , oldValue ) {

if( newValue < 10 ) {

$scope.yanzheng = ‘ 字符不能少于10个 ’

} else {s

$scope.yanzheng = ‘ ’;

}

} )

} ] )

* 创建服务
  + 作用：将公共的代码抽出来
  + 介绍：

1. 服务也是模块下的方法要创建服务先创建模块
2. 在模块对象下有一个**service**方法
3. 第一个参数是服务名称 第二个参数以数组形式存在 和控制器的使用方法一样
4. 将服务对应的js文件在页面中用script标签引入 将服务模块做为主模块的依赖模块
   * 语法： **模块对象.service( ‘ 服务的名字 ’ , [ function ( ) { } ] )**
   * **例 ：** service.js : **angular.module( ‘ myService ’ , [ ] ) . service ( ‘ serviceName ’ , [ function(){**

// angular 将服务处理函数作为构造器来用 会帮我们new这个处理函数 所以绑定属性的时候用 **this**

**this.name = ‘ 张三 ’ ;**

**this.fn =function ( ) {  
 alert( ‘ 我是李四 ’ )**

**}**

**} ] )**

< script src=”./service.js” >< /script > // 引入服务包

< scripr >

angular.module( ‘ myapp ’ , [ myService ] ) // 把服务包作为依赖包

. controller ( ‘ one ’ , [ ‘ $scope ’ , **‘ serviceNmae ’** , functiono( $scope , **serviceNmae** ){

console.log( **serviceNmae.name** ) // 张三

} ] ).controller ( ‘ two ’ , [ ‘ $scope ’ , **‘ serviceNmae ’** , function( $scope , **serviceName** ){

**serviceName.fn()**  // 李四

} ] )

< /script >