# AngularJS1.x快速入门

## 简介

AngularJS是一个基于**MVC处理模式**，实现了**MVVM数据双向绑定**的用于开发动态web项目的JavaScript框架。以其数据和展现分离、MVVM、MVC、DI等强大的特性活跃于前端开发市场，是前端敏捷开发使用的主流的必须掌握的框架之一。

AnuglarJS是**前后端分离软件架构模型**下，前端业务处理的解决方案！

|  |
| --- |
| 教师注意：库 VS框架  **库：**封装原生JavaScript实现的大量函数的集合，是提供了各种特定功能的函数库  通过引入的库中的函数，可以快捷实现一些特定的处理过程，如截取字符串等等  **框架：**一种封装了原生JavaScript实现的函数、功能、组件、处理流程的特殊实现  通过引入框架中的配置，可以实现一定的逻辑处理功能，如Bootstrap中的轮播图等等 |

AngularJS是web应用前端的一个完整的解决方案，框架不仅提供了优雅的DOM操作和基于Ajax的数据交互技术，同时进行了良好的结构整理和常规的业务逻辑处理的封装，使得Angular构建的应用能够灵活的适应不同的项目场景。

|  |
| --- |
| 注意：教师AngularJS中提供了如下的解决方案用以前端开发：  数据双向绑定、基本模板指令、自定义指令、表单验证、路由操作、依赖注入、过滤器、内置服务、自定义服务、组件、模块 |

同样，AngularJS框架从2013年开始在国内出现应用，一直到2016年活跃于web前端项目开发中，丰富的社区资源也是一个非常重要的原因。

|  |
| --- |
| 外链：链接常见AngularJS社区  http://docs.angularjs.org  <http://www.angularjs.net.cn/tutorial/1.html>  <https://github.com/aztack/AngularJS-translation>  <https://github.com/dolymood/learning-angular>  <http://www.angularjs.cn/>  <https://www.zouyesheng.com/angular.html>  <http://blog.darkthread.net/blogs/darkthreadtw/archive/tags/AngularJS/default.aspx>  <http://blog.jobbole.com/52857/>  <http://blog.jobbole.com/62999/> |

|  |
| --- |
| 外链：链接AngularJS1.x官方网站  <http://angularjs.org>  AngularJS1.x代码下载  <http://code.angularjs.org>  AngularJS1.x官方教程  <https://code.angularjs.org/1.6.4/docs/tutorial>  AngularJS1.x官方开发向导  <https://code.angularjs.org/1.6.4/docs/guide>  AngularJS1.x官方API  <https://code.angularjs.org/1.6.4/docs/api> |

## Quick start

**入门程序1：实现一实时提取用户键盘输入内容的记录器**

|  |
| --- |
| 脚印设计思路：传统方式<通过键盘按键事件进行处理>  <input onkeyup=”record()” type=”text”/>  <div>输出记录:<span></span></div>  ..  function record() {..} |

|  |
| --- |
| 脚印代码：angular中，通过数据的双向绑定直接实现  <html ng-app>  <input type=”text” ng-model=”info”>  <div>输出记录：{{info}}</div>  </html> |

**需求案例1：实现一个购物车中更新数据的功能**

功能要求：修改购买数量，在修改的同时，商品的小计购买金额发生变化。

|  |
| --- |
| 脚印代码：  <div ng-init=”price=12;count=1”>  <input type=”text” ng-model=”count”>  <div>小计金额：{{price \* count}}</div>  </div> |

AngularJS应用中，对传统的HTML进行了扩展，增加并强化了HTML的功能，从入口属性ng-app(指令)开始加载应用，并逐行编辑解释这些扩展的内容。

遇到input标签中的属性ng-model(指令)，就会将它的值当成一个变量进行处理

遇到{{}}这样的结构，就会将括号中的值当成表达式进行处理

|  |
| --- |
| 注意：mus教师tache语法，通过两个大括号来包含一个表达式的结构  Angular会对mustache语法进行编译并解释成可以运算的表达式进行处理 |

AngularJS就是因为对传统的HTML进行了功能的扩展，所以在一定程度上，实现了软件开发过程中前后端的架构级别的分离——也就是软件**前后端分离架构**模型。

同时将前端的内容展示、业务功能处理、数据交互有组织的进行了规划，让前端HTML更加适合如今的企业级平台的web软件开发。

|  |
| --- |
| 原子扩展：传统软件模型是前后端耦合架构模式，所谓耦合架构就是服务器后端和前端紧密结合，这样的架构模型下HTML是不满足web开发需要的。如JavaEE开发的web软件，服务器后端使用java语言以及它的框架来完成开发，前端使用JSP进行开发，JSP是对HTML的扩展，可以在HTML中添加java代码和java定义的标签，但是JSP是只能用于Java后台的视图。PHP开发的服务器后台，页面部分使用PHP进行开发，PHP文件也是在HTML中进行了大量的功能扩展。  AngularJS类似的框架出现之后，对于前端进行了大量的封装和扩展，让HTML能更加友好的支持后端主流的语言并协同进行web软件的开发，也就是前后端开始分离，不再像以前那样紧密的耦合。 |

## 专业术语概述

|  |  |
| --- | --- |
| **术语** | **描述** |
| 模板template | 包含了Angular特殊扩展标记的HTML代码 |
| 指令directive | 扩展的HTML代码，自定义的标签、属性等等 |
| 模型model | 保存在JS中，用来和用户交互的数据 |
| 作用域scope | 模型数据在HTML页面中产生作用的范围 |
| 表达式expression | AngularJS在HTML页面中可以运算的语法 |
| 编译器compiler | 被用来解释HTML代码中的Angular代码 |
| 过滤器filter | 对HTML页面中输出的数据进行指定格式展示 |
| 视图view | 统称用户看到的HTML视图页面 |
| 数据绑定data binding | JS中特定的数据和HTML页面上的数据的关联关系 |
| 控制器controller | 给视图view提供功能支持的东东 |
| 依赖注入  dependency injection | Angular自动创建对象并传递对象的一种方式 |
| 注入器  injector | 专门用来实现依赖注入（DI）的容器 |
| 模块module | Angular用来对项目进行最高层次封装的东东 |
| 服务service | Angular用来对视图view提供业务功能支持的东东 |
| 组件component | 用于对网页的公共部分进行封装重用的Angular代码  通常会包含模板、指令、服务等等 |

## 引导程序

任何应用程序，运行过程中都会有一个运行的入口。

AngularJS应用程序也是一样，正常情况下，引导Ag应用的入口程序有两种：

* ng-app指令：店家推荐，项目中常用的方式
* bootstrap手工引导

常规模式下，是通过ng-app指令引导Angular应用进行工作的。

网页中引入Angular之后，程序运行时会自动查找ng-app指令并从这个入口开始加载编译并解释执行。

|  |
| --- |
| **脚印代码**：通常，页面中会定义ng-app熟悉来指定入口  在script中通过angular的module来加载ng-app指定的模块  <html ng-app=”myApp”>  <script>  angular.module(“myApp”, []);  </script>  </html> |

|  |
| --- |
| **脚印代码**：某些特殊情况下，也可以通过脚本来动态控制加载启动Angular应用  通过angular.bootstrap(dom, module)来启动应用  <html>  <div id=”myDiv”></div>  <script>  angular.module(“myApp”, []);  var \_div = document.getElementById(“myDiv”);  angular.bootstrap(\_div, “myApp”);  </script>  </html> |
|  |

## 基础语法结构

开始深入学习AngularJS之前，先简单认识AngularJS中的各个部分都是怎么定义和使用的，方便解决后续学习过程中的一些困扰。

|  |
| --- |
| 讲师ng-app |

Angular应用运行的入口指令，常规情况下，是写在我们前端项目的入口文件的根标签上的，用于在项目启动的时候引导Angular应用

|  |
| --- |
| 讲师angular.module(..) |

Angular是通过模块来管理我们前端项目中的数据的，通常情况下一个项目中会包含各种各样的数据，如管理员、会员、商品列表、商品、新闻列表、新闻、公告等等

模块时需要在应用启动的时候就需要加载的，所以Angular在设计的过程中对于入口指令进行了改造可以绑定一个值，这个值就是系统的主模块

|  |
| --- |
| 基本指令讲师 |

指令是Angular中使用比较多的一个部分，Angular中的内置指令都是ng-开头的一种特殊的语法结构，如:ng-app等，常见的基本指令如下：

**ng-app=”myApp”:程序运行的入口，通过名称绑定一个模块**

ng-init:用于在程序运行的过程中，初始化一些变量的数据的操作

**ng-model=”param”:数据绑定的指令，主要用于表单元素上的数据绑定**

**ng-bind:数据绑定的指令，用于将变量的数据显示到页面上，用于替代mustache语法的**

**ng-controller:控制器指令，用于在页面中指定控制器作用范围，通常作为属性出现**

**ng-[event]:事件指令，用于在页面中发生某些事件时调用指令的函数**

|  |
| --- |
| 讲师基本语法 |

var app = angular.module(“name”, []):用于定义一个angular的模块

* name是模块的名称
* []中可以添加其他的模块

app.controller(“name”, function() {}):用于定义一个angular的控制器

* name是控制器的名称
* function(){}是这个控制器要处理功能的函数

|  |
| --- |
| 讲师$scope作用域 |

$scope是放在控制器函数中的一个参数。这个参数不需要传值，而是angular自动赋值

在操作的过程中$scope就相当于一个容器，可以在$scope上声明变量或者函数，$scope上的变量和函数会自动和视图页面中的变量进行绑定，称为$scope挂载数据。

|  |
| --- |
| 讲师$rootScope作用域 |

$rootScope是AngularJS的根作用域，也是全局作用域，放在控制器函数中的一个参数，参数不需要传值，angular会自动赋值，挂载在$rootScope上的数据，会被Angular应用中的所有模块下的子模块和控制器公用。

|  |
| --- |
| 讲师run()函数 |

配合$rootScope使用，用于声明和初始化全局数据【当前模块下所有的组件可以访问的数据】

app.run(function($rootScope) { // 处理代码})

|  |
| --- |
| 讲师$scope.$watch() |

挂载在$scope上，用于监控数据的变化

监控单个变量：$scope.$watch(“name”, function() {// 处理代码})

监控多个变量：$scope.$watch(“name1 + name2 + ..” + function() {})

|  |
| --- |
| angular事件处理讲师 |

AngularJS中的事件操作，由于它自己虚拟DOM结构所以不支持普通事件，通常通过它自己的事件指令来调用通过$scope挂在到控制器中的函数执行完成，事件指令其实就是对于常见事件的封装，以ng-开头，加上事件名称即可，如鼠标单击事件ng-click

事件ng-click=”fn”，这里的fn必须是挂载在$scope作用域上的函数。

## MVC处理模式

**MVC不是一个技术，是一种处理问题的思路，也就是编程思想。**

**这种编程思想的核心是将数据和展示分离，**通过控制器挂载并进行一定的功能逻辑的处理以达到让代码具备强大的扩展性的目的。

MVC中主要包含三个模块

* View:视图——用于数据的展示和交互
* Model:数据模型——用与数据的封装和运算
* Controller:控制器——用于对视图和数据模型之间的联系进行逻辑关系的处理

|  |
| --- |
| 原子扩展：MVC和MVP  MVC是传统的处理模式，但是随着近年来前端技术发展迅速，数据双向绑定处理模式的出现，传统的MVC也进行了细化，出现了MVP处理模式  **MVC:**视图view和数据模型model的数据是双向绑定的，控制器只是作为提供逻辑处理服务的中间件  **MVP:**视图view和数据模型model的数据是单向绑定的，控制器作为视图view和数据模型model之间传递数据和逻辑处理的桥梁  常规情况下，由于MVC处理模式下数据双向绑定造成的数据源的污染，以及view对于model严重的依赖性，所以建议在mvc处理模式下，复杂业务的数据的传递和逻辑处理过程还是通过控制器进行，而不是直接绑定 |

## 控制器<controller>

AngularJS1.x的核心之一就是控制器部分，主要对于视图中的数据和事件处理函数进行挂载，同时进行一定的业务功能的底层封装和处理。

|  |
| --- |
| 讲师控制器使用注意事项 |

**控制器在AngularJS中的作用**

* 通过$scope进行数据状态的初始化操作
* 通过$scope进行事件处理函数的挂载操作

**注意：不要使用控制器做下面的事情**

* DOM操作：使用AngularJS中的**数据双向绑定**和**自定义指令/组件**执行操作
* 表单处理：使用Angular中的form controls进行操作
* 数据格式化展示：使用Angular中的过滤器Filter来进行操作
* 不同控制器之间的数据共享：使用Angular中的自定义服务Service进行处理
* 组件生命周期的操作：使用Angular中的自定义服务Service进行处理

|  |
| --- |
| 脚印代码：控制器语法结构  var app = angular.module(“myApp”, []);  app.controller(“myCtrl”, function($scope) {  // 控制器函数操作部分  // 控制器主要进行数据的初始化操作和事件函数的定义  }); |

### $scope作用域

$scope作为控制器函数一个非常重要的参数，AngularJS通过依赖注入的方式进行自动赋值，挂载数据完成控制器中的数据处理。每个$scope都拥有自己控制器的作用域，并且都独立于当前的控制器。

|  |
| --- |
| 讲师$scope挂载变量 |

通过控制器函数中的$scope参数，直接将JS变量添加到$scope对象上。

$scope.name = “jerry”;

这里name是变量名称，jerry是变量值；变量值可以是字面量直接赋值，也可以是后端接口获取数据之后进行赋值

|  |
| --- |
| 讲师$scope挂载事件处理函数/函数 |

通过Angular的事件指令，在指定的DOM元素上挂载事件，然后在$scope作用域对象上挂载事件处理函数，完成事件的完整流程操作。

<div ng-click=”clickMe()”>点击我</div>

$scope.clickMe = function() {..}

补充：可以通过$event来传递一个事件对象，方便在Angular中通过事件对象进行后续更多功能的操作。

<div ng-click=”clickMe($event)”>点击我</div>

|  |
| --- |
| 讲师$scope作用域范围 |

$scope上挂载的数据，和控制器controller进行了绑定；控制器controller和网页上的DOM元素进行了绑定，网页中DOM元素的开始标签和结束标签范围，就是控制器的作用范围，也是$scope上挂载的数据的作用范围.

|  |
| --- |
| 讲师$scope.$watch(expression, listener, [objectEquality]) |

$watch()是$scope对象的一个函数，该概述用于在代码执行的过程中对变量的数据进行监听，一旦变量的内容发生变动就执行对应的函数操作。

* **函数第一个参数是监听表达式，如果是单个变量直接写变量名称；**

$scope.price = 12;

$scope.$watch(“x”, function() { // TODO; })

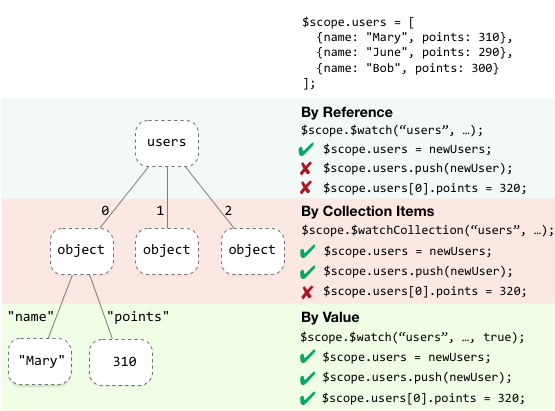
* **如果是多个变量，监听的多个变量中间通过+连接符连接**

$scope.price = 12;

$scope.count = 1;

$scope.$watch(“price + count”, function() { // TODO; });

* **如果监听的是数组集合内部的数据，需要使用$watchCollection(..)进行监听**
* **如果监听的是对象内部的属性数据，需要将$watch()的第三个参数设置为true**



|  |
| --- |
| 讲师$scope.$apply(); |

某些情况下，页面的数据在特定的条件下，执行了延时/异步操作，此时页面上的数据和对应的变量的绑定状态遭到了一定程度的破坏，就需要进行代码级别的主动的干预进行数据的推送来完成数据的同步

$scope.data = “hello”;

setTimeout(function() {

$scope.data = “new data changed!”

**$scope.apply();**

}, 1000);

### $rootScope作用域

每一个AngularJS应用，都有唯一的一个全局作用域范围，也称为根作用域，Angular中其他的作用域都是这个根作用域的后代/子作用域。

AngularJS的根作用域/全局作用域：$rootScope

根作用域使用时，通过注入的方式添加到控制器处理函数的入参中。

|  |
| --- |
| 讲师angular.module.run(fn); |

在AngularJS中，有一个特殊的函数run()，用于进行全局的数据处理，结合$rootScope一起用来初始化全局数据，这些数据可以被模块下所有的子模块和控制器共享。

### 依赖注入

AngularJS的控制器我们有了简单的初步的认识，如果处理函数中需要添加参数依赖，在函数中直接写上依赖对应的名称即可。但是，在前端自动化构建时，部分构建工具会对函数的参数进行优化处理，造成Angular中处理函数失效的问题。

AngularJS提供了一种依赖注入的方式来解决上述问题，依赖注入是从后端语言参考的一种编程的设计模式，通过将当前代码块运行依赖的模块进行注入完成后续的处理操作。

app.controller(“myCtrl”, **[“$scope”, function(s)** {..}]);

上面的代码中s就被注入了$scope对象，在函数中，s就可以被当成$scope进行使用。

|  |
| --- |
| 教师注意：进行依赖注入操作时，方括号中的字符串表示的参数列表和函数中的参数顺序一定要保持一致，AngularJS会根据顺序进行赋值！ |

### 全局控制器和局部控制器

AngularJS1.2版本之前，AngularJS中定义控制器，是直接定义成window全局函数进行处理的，这样的控制器可以被所有的模块共用，称为全局控制器；这样的处理模式的优点是编写方便，处理流程清晰；缺点是容易造成变量和函数的全局污染。所以AngularJS在1.2版本之后，进行了优化调整，废弃了全局控制器，而是将控制器注册在模块中，成为模块中的局部控制器存在，局部控制器也就相当于限定了控制器的作用域范围。

|  |
| --- |
| 教师注意：控制器的改进  在Angular1.2版本之前，控制器是通过静态函数直接定义的，并且控制器中的变量是作用于全局的。但是全局控制器的作用域造成了控制器数据的全局污染。所以在AngularJS1.2版本以后，进行了更新，不再添加任何全局控制器，而是将控制器挂载到模块上注册成为局部作用域，方便数据的高可控性和可操作性。 |

### 控制器嵌套

AngularJS中的控制器是可以进行嵌套的，控制器本身作为HTML的一种扩展实现，和HTML的DOM元素绑定，作用域范围依附于DOM结构。

## 组件化<component>

AngularJS作为一个模块化组件开发的web前端框架，对于视图进行组件化封装以达到重用的目的，是它非常优秀的一个亮点。

什么是组件化？组件化是将页面中一部分UI封装起来进行重复使用，UI中的数据可以是不同的，甚至组件中可以对UI进行适当的业务逻辑处理，如链接跳转、数据运算等等

|  |
| --- |
| 脚印代码：组件定义语法结构    var app = angular.module(“myApp”, []);  app.component(“name”, {  settings  }); |

组件选项：

* template:组件替换模板字符串
* templateUrl:组件替换html模板页面
* controller:组件控制器

|  |
| --- |
| 脚印代码：组件定义语法结构    var app = angular.module(“myApp”, []);  app.component(“myComp”, {  template:”<h1>组件化开发</h1>”,  controller:function($scope) {  控制器处理代码  }  }); |

|  |
| --- |
| 教师注意：组件命名规则  组件的命名有框架自己的规则和使用规范，都是根据实际项目需要进行封装处理的  组件命名规则：驼峰命名法，见名知意，如：myComp  使用规则：W3C命名规则，将大写字母转换成小写字母，前面加中划线，如<my-comp></my-comp> |

## 项目文件结构管理

使用AngularJS进行文件的模块化开发之后，前端项目网页就被拆分成不同的模块进行管理，不同的模块之间进行整合完成项目页面的快速搭建。

|  |
| --- |
| |-- project/  |-- images/  |-- css/  |-- js/  |-- lib/  |-- third library/  |-- app/  |-- modules/  |-- app.module.js  |-- index.html |

## 指令 & 自定义指令<directive>

指令是AngularJS对HTML的功能的扩展，增加了额外的标签或者属性来实现了更加强大的数据处理功能。

AngularJS中提供了内置指令以方便开发人员对于常见功能的处理，同时也可以通过自定义指令的方式来实现更加完善和强大的功能处理；内置的指令都是ng-开头的，常见的一般有控制指令如np-app等、渲染指令如ng-repeat等、节点指令如ng-class等等、事件处理指令如ng-click等等；

### 内置指令

|  |
| --- |
| 讲师控制指令 |

ng-app:用于指定AngularJS应用启动的入口和挂载应用的根模块

ng-controller:用于指定AngularJS模块上的控制器和对应的DOM元素的绑定

|  |
| --- |
| 渲染指令讲师 |

ng-repeat:用于循环输出指定的数据

ng-bind:用于展示绑定的变量中的数据

ng-bind-html:已浏览器可以解释的方式进行数据的展示

ng-include:页面包含，可以通过该指令将指定的另一个页面包含到当前页面中

|  |
| --- |
| 节点指令讲师 |

ng-class:用于通过双向绑定的数据变量来动态控制DOM元素的Class样式

ng-class-even/odd:用于和ng-repeat结合使用时，控制偶数行和奇数行不同的样式处理，添加不同的样式进行隔行样式分离

ng-style:用于直接给Html标签添加行内样式的处理，样式按照css标准语法添加

ng-src:用于将图片路径动态渲染到img标签中

ng-href:用于将超链接路径动态渲染到a标签中

ng-if:选择指令，用于根据指定条件进行DOM元素的移除或者重绘到DOM结构的操作

ng-show:显示指令，用于当条件为true时将指定的元素显示绘制到网页中

ng-hide:隐藏指令，用于当条件为true时将指定的元素在页面上隐藏display:none

ng-switch:多条件选择指令，配合子指令ng-switch-when和ng-switch-default一起使用，来进行多条件下的选择输出

|  |
| --- |
| 事件指令讲师 |

ng-click:用于捕捉用户鼠标单击操作，执行指定的事件处理函数

ng-dbclick:鼠标双击

ng-keydown:键盘按键按下

ng-keyup:键盘按键抬起

ng-mousedown:鼠标左键按下

ng-mouseup:鼠标左键抬起

.. .. ..

* 注意：事件处理的过程中，可以通过$event将事件对象传递给处理函数，通过事件对对象可以进行阻止事件冒泡、阻止浏览器默认行为、获取鼠标位置等等各种原生操作。

### 自定义指令

TODO

## 过滤器 & 自定义过滤器

AngularJS针对我们在视图页面上输出和展示的数据，为了方便展示的风格更加契合用户的需求并提高开发效率，通过过滤器对视图页面上展示的数据进行了二次处理，但是实际需求是千变万化的，Angular针对性的将一部分项目中常见的功能抽取出来并封装为内置的过滤器，同时也提供了接口，方便开发人员可以自定义过滤器来实现个性化的需求。

### 内置过滤器

|  |
| --- |
| 讲师货币过滤器currency |

描述：货币过滤器将用于展示的数字数据，转换成对应的货币格式，默认是$

语法：{{currency\_expression | currency:f:n }}

解释：currency将前面的数字按照货币格式展示

参数f用于指定自定义符号，默认$

参数n用于指定保留的小数位数，默认2位

|  |
| --- |
| 讲师日期过滤器date |

描述：日期过滤器用于将日期时间快捷的按照指定的格式进行输出展示

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

解释：y/M/d/h/m/s/E分别表示年/月/日/时/分/秒/星期

|  |
| --- |
| 筛选匹配过滤器讲师filter |

描述：筛选匹配过滤器用于将展示的数据中查询筛选关键词信息，符合条件的予以展示

语法：{{array | filter:”keyword”}}

|  |
| --- |
| 讲师JSON过滤器json |

描述：该过滤器用于将一个JS对象格式化为JSON字符串，官方推荐主要用于**程序调试**

语法：{{jsJson | json}}

|  |
| --- |
| 长度限制过滤器：讲师limitTo |

描述：该过滤器用于限制数组或者字符串等输出的长度

语法：{{arr | limitTo : limit : begin}}

解释：limit参数表示限制的长度

begin参数表示从哪个位置开始过滤，默认从开始位置过滤

|  |
| --- |
| 讲师小写转换过滤器lowercase |

描述：该过滤器用于将展示的英文字符全部转换成小写

语法：{{expression | lowercase}}

|  |
| --- |
| 大写转换过滤器讲师uppercase |

描述：该过滤器用于将展示的英文字符全部转换成大写

语法：{{expression | uppercase}}

|  |
| --- |
| 数字格式化过滤器讲师number |

描述：该过滤器将数字转换成标准千位分隔的形式

语法：{{num | number : n}}

解释：number将num数字添加千位分隔符，同时参数n用于保留小数位数

|  |
| --- |
| 排序过滤器讲师orderBy |

描述：该过滤器用于将一个数组中的元素进行排序，接受一个参数用来指定排序规则

语法：{{arr | orderBy:param:ot}}

解释：arr中正常展示的数据，按照指定的属性param的值进行排序

参数param用于指定排序规则，通常是arr中的一个属性

参数ot用于指定排序顺序，默认false[顺序排列]，true[倒序排列]

### 自定义过滤器

针对前端数据展示各种个性化的需求，Angualr提供了可以让开发人员进行自定义过滤器的方式来完成个性化的格式化展示。

|  |
| --- |
| 语法结构讲师 |

AngularJS通过angular.module(.).filter(name, fn)函数来自定义过滤器

* name参数用于指定自定义过滤器的名称
* fn参数用于指定过滤器的处理函数

fn处理函数中，通过返回一个闭包函数来完成过滤器的处理功能

function() {

return function(value, [args]){

// 处理代码

return value;

}

}

* value就是自定义过滤器接收到的要格式化处理的数据
* args表示接收到的自定义过滤器的附带参数
* 必须在闭包函数中通过return value;将处理结果的数据返回用于展示

## 服务 & 自定义服务<service>

Angular为了更好的使用数据来服务前端页面，提供了服务这样一个组件来进行处理，服务在Angular中充当了业务提供者的角色，主要用于服务控制器之间以及前端和后端交互的各个方面。

|  |
| --- |
| Angular services are substitutable objects that are wired together using [dependency injection (DI)](https://code.angularjs.org/1.6.4/docs/guide/di). You can use services to organize and share code across your app. |

AngularJS官方将服务作为一个对象存在，在使用的时候通过依赖注入的方式，类似于前文中使用的$scope就是一个数据挂载的作用域服务对象，通过依赖注入的方式使用，使用服务来进行业务处理和在整个应用中进行数据的互相访问。

|  |
| --- |
| Angular services are:   * Lazily instantiated – Angular only instantiates a service when an application component depends on it. * Singletons – Each component dependent on a service gets a reference to the single instance generated by the service factory. |

同时服务也有自己的特点，懒加载~Angular的服务只有在应用依赖指定服务的时候才会实例化这个服务对象，并且服务对象在整个应用中是只有一个实例的，任何组件在使用服务的时候都是通过这个实例对象的引用来使用它的。

Angular中的服务主要有两种，自定义服务和内置服务。自定义服务用于特定的项目中进行个性化业务处理过程，内置服务用于一些特定的公共的功能提供的，如异步数据请求等等

### 自定义服务

|  |
| --- |
| 讲师自定义服务是Angular中提供的一个非常重要的功能，从有了前后端分离的软件结构之后，前端就负责了一部分业务逻辑的处理过程以及和后端的交互过程。控制器主要作为数据和函数挂载的作用来存在的，在这样的模式下，数据~也就是从后端访问数据和对数据的格式化处理过程以及验证等等操作，都交由服务来进行。 |

Angular中的自定义服务的实现方式有三种:factory、provide、service

|  |
| --- |
| 脚印自定义服务：factory  factory方式实现的自定义服务在项目中使用较为频繁，以其简单的语法和使用方式流行  var app = angular.module(“myApp”, []);  app.controller(“myCtrl”, function($scope, myService) {  $scope.value = myService;  });  app.factory(“myService”, function() {  return “jerry”;  }); |

|  |
| --- |
| 脚印自定义服务：service  service方式实现的自定义服务，返回的数据必须是JSON对象，在某些项目中为了规范数据的返回值，约定统一使用对象进行处理。service就比较符合这样的规范，所以使用较为频繁  app.service(“myService”, function() {  return {name:”tom”};  }); |

|  |
| --- |
| 脚印自定义服务：provider  provider自定义服务，在angular框架内部使用较多，由于代码操作更加趋于底层造成代码可读性的降低，对于常规项目开发有一定的难度，所以项目开发使用较少  app.provider(“myServiceProvider”, function() {  this.$get(function() {  return value;  });  }); |

AngularJS中的自定义服务一旦创建好，只有在其他的模块依赖的时候才会被应用加载初始化，主要在AngualrJS的指令、控制器、过滤器、组件等等这些东西中注入使用！

### 内置服务

AngularJS提供了大量的内置服务，通过这些内置服务就可以快捷的进行一些业务功能流程的自动处理了，如：

$window用于注入原生JS中的window对象

$document用于注入原生JS中的document文档对象

$timeout用于注入封装的原生JS中的setTimeout()函数处理过程

$interval用于注入封装的原生JS中的setInterval()函数处理过程

$location用于注入原生JS中的location对象方便对于URL地址等操作

$http用于注入封装的Ajax操作进行异步数据请求等等

|  |
| --- |
| 内置服务：讲师$http |

$http服务是AngularJS的核心服务之一，这个服务主要封装了XMLHttpRequest对象和JSONP数据访问模式来完成远程服务的数据请求！

**常规语法结构：**

$http({

method:”GET”,/\* 请求发送方式 \*/

[url:”http://......../com” /\*](url:) 请求地址\*/

}).then( /\* then()函数表示请求完成之后的操作 \*/

function(response) {

/\* 请求成功之后的操作函数 \*/

},

function(response) {

/\* 请求失败时候的操作函数 \*/

}

);

AngularJS为了方便开发人员进行快捷的异步数据请求的处理，提供了一系列的快捷函数方便开发，主要由如下的函数：

* $http.get()
* $http.post()
* $http.jsonp()
* $http.header()
* $http.patch()
* $http.put()
* $http.delete()

$http.get(“url”).then(fn1, fn2);

其实快捷方式，就是简单的封装了$http服务，常规项目开发时，**使用较多的是原始的内置服务$http的$http({}).then(fn1, fn2)函数进行异步数据处理。**

## 路由<ng-router & ui-router>

路由是AngularJS构建单页面应用的基础。

单页面应用，也称为SPA，是singleton page application的缩写。

|  |
| --- |
| 呼叫中心单页面应用：程序在执行的过程中，我们不论访问任何链接，都会在同一个页面中展示数据  在整个项目执行过程中，客户看到的都是唯一的一个页面，不同的是~在整个页面中，会根据用户请求的内容的不同，动态的更新页面中的某一个部分的数据，在这样的模式下，项目运行只需要一个基础的页面即可，其他的都是模板。模板就是用于在这个基础页面中更新的数据。 |

传统项目中，通过大量的页面之间的跳转，来实现不同的功能的业务处理，如index.html主要用于系统首页数据的展示、login.html页面主要用于用户登录业务的处理，而相对应的regist.html页面主要用于新用户的注册业务的处理，一个项目中会充斥着大量的页面和页面之间的跳转处理，如果项目的业务非常复杂的情况下，网页的开发和维护就会造成大量重复的工作，同时也提升了开发和维护的复杂度。

SPA通过路由功能，让我们的WEB应用，在运行过程中，依托于某一个页面模板进行业务处理，在SPA应用中，我们可以通过一个模板页面和其他的业务页面进行不同的路由组合来完成复杂和庞大的页面逻辑和业务的处理！

|  |
| --- |
| 呼叫中心什么是路由？  路由，就是网络数据或者请求进行分发的一个网络组件。  路由就是一个用于请求URL分发和跳转的一个应用组件，Angular中通过$routeProvider路由服务提供者进行核心的配置处理。 |

### ng路由

ng路由是AngularJS官方提供的一种简单的路由操作。

ng路由主要分三个组成部分：路由指令、路由服务、路由服务提供者

|  |
| --- |
| 路由指令讲师:ng-view |

**描述：**

ngView指令主要用于将路由指向的页面渲染到当前页面的布局中。

**指令概述：**

指令会创建自己独立的作用域

**语法：**

<ng-view [onload=”string”] [autoscroll]=”string”></ng-view>

onload:当视图发生改变时执行属性值中的表达式

autoscroll:当视图发生改变时自动触发$anchorScroll事件

**事件：**

路由视图一旦加载时，就会自动触发$viewContentLoaded事件

|  |
| --- |
| 路由讲师提供者:$routeProvider |

**描述：**

内置服务对象$routeProvider主要用于进行路由配置

该服务的使用必须依赖ngRoute模块，也就是项目中必须添加animate-route.js

**主要方法：**

when(path, route);用于在访问path路径时，跳转到route指定的视图

|-- path：路由跳转的路径

|-- route：路由对象<组件对象>，一个JSON对象

otherwise(params);用于在访问不存在的路径时，跳转的默认路径或者视图

|-- params：指定路径或者路由对象

|  |
| --- |
| 内置服务讲师<路由服务>:$route & $routeParams |

$route服务被用于进行深层超链接信息的描述，它会监听$location.url()地址并进行url地址和指定的路由视图之间的映射关系。

$routeParams服务允许开发人员可以进行路由中参数的处理。

### ui路由

TODO

## 模块化开发<module>

TODO

## 表单处理<form>

TODO

## 动画处理<annimation>

TODO

## 项目开发全流程分析

TODO

|  |
| --- |
| 标题讲师 |

|  |
| --- |
| 提示：呼叫中心 |

|  |
| --- |
| 下载：从云中下载 |

|  |
| --- |
| 注意：教师 |

|  |
| --- |
| 外链：链接 |

|  |
| --- |
| 原子扩展： |

|  |
| --- |
| 脚印代码： |