# Mv\*

MVx——数据、表现分离

MVC

Model 模型---数据

View 视图----表现层，html,template

Controller 控制器------业务逻辑

# angularJs

Angular版本下载：<https://code.angularjs.org/>

Angular是什么，干什么？

减轻程序员负担，

**作为MV\*框架，把数据、表现、逻辑分离开**

**兼容性：高版本浏览器，IE67不兼容**

**适合性：MV框架，数据参与项目，数据交互**

版本：1.3版本前后有很大差别

* 1.5.8
* 1.2.3

## 属性：

$scope 作用域，数据 ng-init添加在 $scope上

## 扩展了html功能

扩展了html功能

颠覆性，可扩展

Ng-app=’’ 圈定范围

## 双向数据绑定、

对于可输入的标签，如input,textarea

## 依赖注入

比如：去饭店--正常，入座拿菜单点菜

函数：函数的实参参数，由调用方决定

依赖注入的情况下——参数由定义方决定。

## 模板

{{ 数据 }}

1. angular 与 javaScript 不互通。【正常情况下（函数、变量）】
2. Angular的开发和传统开发模式完全不同--只需要盯住数据。

## 互通

Controller

## Angular接管了UI

# 命令

## 自定义指令——组件化

App.directive(name,function(){

Var json ={

Restrict: ‘E’ //约束,指令的使用作用范围

//E：element元素 C:class类 A:attribute 属性 M:comment注释

Template:’’ 模板，

Replace：true ， //在M范围内必须使用

Transclude : true //嵌入 模板内添加占位符 <ng-transclude></ng-transclude>

}

Return json；

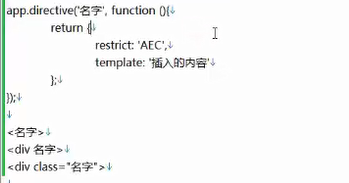
})

<name></name>

<div class=’name’></div>

<div name=’’></div>

<!-- directive: name --> 必须带空格



## 属性指令

* Ng-app = ’ ’ 范围 angular.module(‘app-name’[])
* Ng-init =’a=12 ’ 初始化数据 app.control(‘ctrl-name’,function($scope,$http){})
* Ng-model=’ ’ 模型，数据源，具备双向绑定
* Ng-bind=’ ’ 输出
* Ng-bind-html =’’
* Ng-repeat =”(index, val ) in xx track by $index” 循环输出 track by

**与事件、赋值不能见面**

* + 单一循环
  + 多元素循环
* Ng-src = ‘’ url
* Ng-href =’’ url
* Ng-hide=’’ true/false
* Ng-show=’’true/false
* Ng-clack 延迟加载
* Ng-include =’ a.html’

## 事件

* Ng-click <---> ng-repeat 冲突 ng-click 不能直接赋值，使用函数

# 控制器

逻辑代码应放在控制器中

1. 放js代码
2. 桥梁, 链接

# 模块化---配置ng-

Ng-app 即一个模块

Ng-controller 一个控制域

Angular在开发过程中一切的表现基于数据，在项目开发过程中按照功能进行。这样，在不同的区域、页面，按需加载，最大程度的做到代码重用。同react的模块化。

# ng-依赖注入app.---

App.controller(‘ctrl’ ,function($scope){ });

App.controller(‘ctrl’ ,[‘$scope’ ,function($s){ }]);

## Ng的模块 $

Var app = angular.module(name,[]);

App.xxx(name,function(){

功能

})；

Xxx

### 控制器 contorlller

App.controller ，

### 过滤器 filter

App.filter

### 常量 constant

App.constant(‘VERSION,’xxxx’)

### 变量 value

App.value(name,value);

### 装饰-decorate

App.decorator(‘name’, function($delegate){

$delegate.xx =xxx; 修改的原始name依赖

Return $delegate;

})

### 自带的模块

* Ng-app
* Ng-controller
* $scope 作用域
  + $watch( ele , fn ,true/false); 监视变化/深度监视 $scope.$watch()
  + $apply() 强制更新[脱离angular的场景应用] $scope.$apply()
* $http
  + $get(url,{params:{}}).success(fn(res){}).error(fn(res){});
  + $post(url,{params:{}}).success(fn(res){}).error(fn(res){});
  + $jsonp(url,{params:{}}).success(fn(res){}).error(fn(res){});

脱离了angular的范围，失去作用域，使用$apply()来强制更新。

* $interal() Var timer = $interval(fn,time) $interval.cancel(xxx)
* $timeout() Var timer = $timeout(fn,time) $timeout.cancel(xxx)

### 引入

# 过滤器

## 7.1 自带过滤器

Currency 货币 $

Data

Json

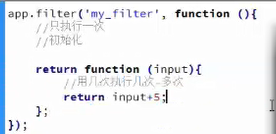
Number

Filter

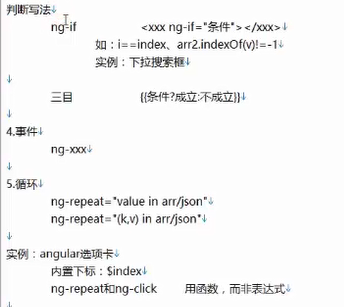
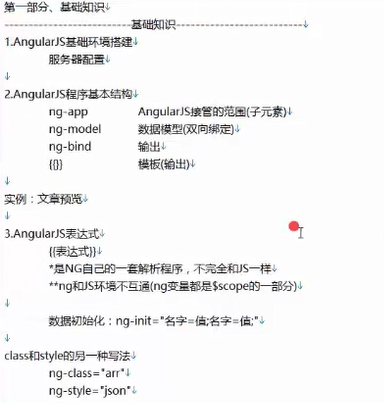
limitTo

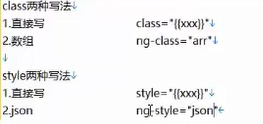
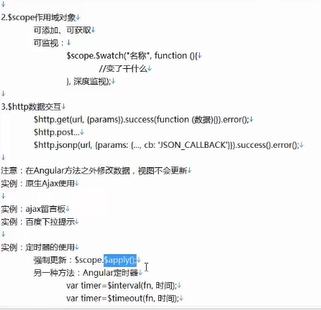
orderBy

## 7.2 自定义过滤器



App.filter(name,fn(){ return function(input){return xx} });







# 事件机制



# 数据共享-消息机制

## 父子级的controller

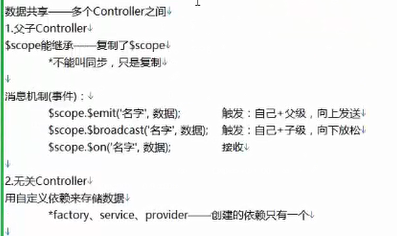
发送：$scope.$emit(‘name’,datas)； 向上发送

广播：$scope.$broadcast(‘name’,datas); 发给自己和自己

**必须在自己建立了on之后才可以发送**

接收：$scope.$on(‘name’,fn(event,datas));

## 无关的Controller



# Router

**使用Hash改变**

引入router模块 ‘ngRoute’

添加 $router

配置route

App.config(function($routeProvider){

$routeProvider.when(

‘url’,

{

Template:

}

)

})

# 延迟加载

Promise请求序列化

Promise.then().then().....

Ajax请求

1 请求数据

2 过滤

3 展现

Ajax(‘url ’,{

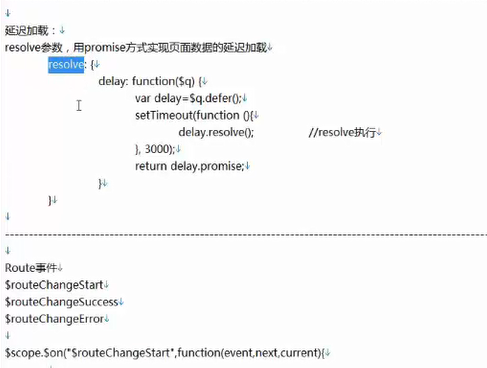
Success:function(){},

For(){},

Ajax() ....

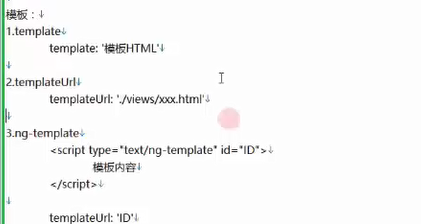
}

## route





### 模板的写法



# 模块化开发

Angular.module(‘name’,[])

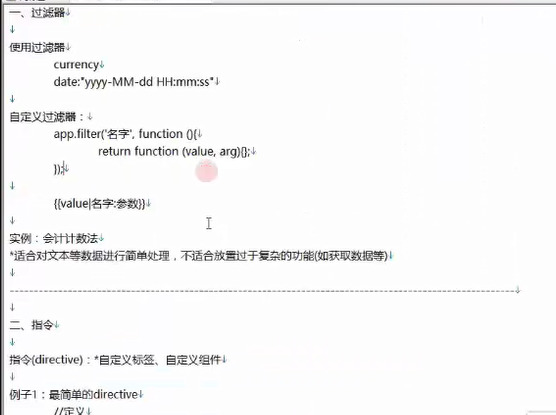
.controller()

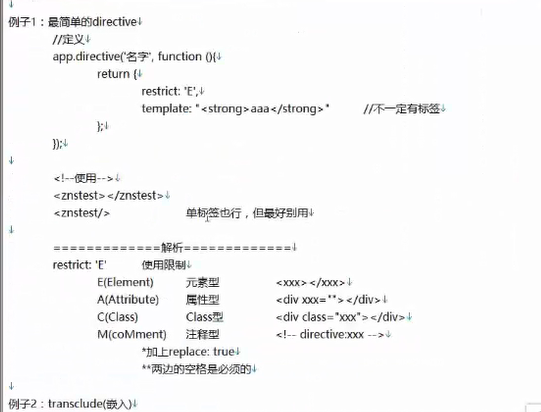
.filter()

1页面引用：

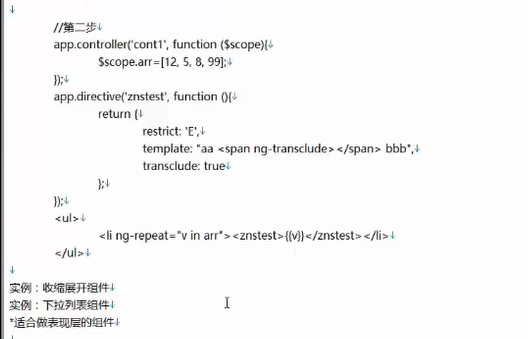
Ng-app=’name’;

2模块依赖

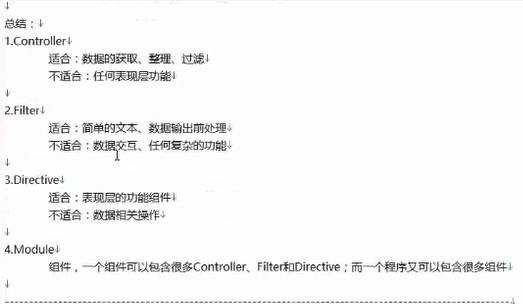












## 自定义依赖注入

### 创建依赖

1 App.factory(‘name’,function(){ return xxxx });

**返回值及名字**

**2 provider(官方推荐)--提供，供应**

App.provider(‘name’,function(){ this.$get=function(){} })

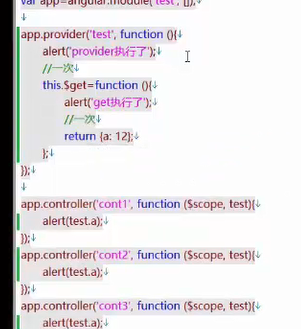
3 service-服务

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

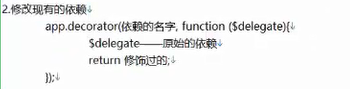
this.lee = 'leeSe'; });

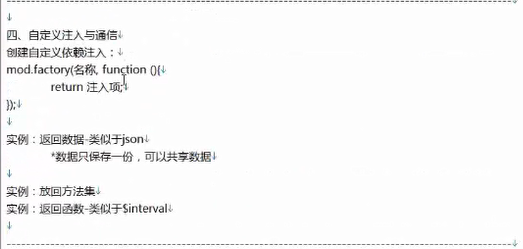
4. constant 常量，不可修饰

1. value 变量

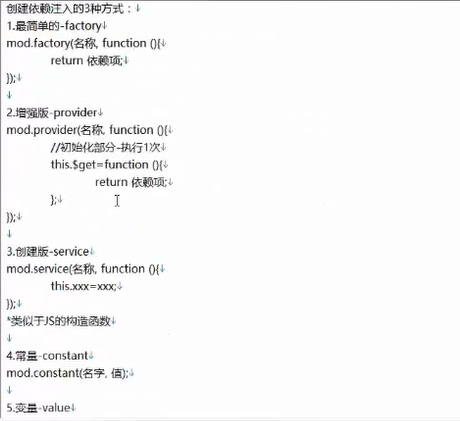


### 修改依赖















Controller 数据操作

Filter 过滤：文字处理

Directive 表现层的组件，【选项卡，显示更多】

Route 分离试图，控制器，路由

Module