**{{}}** 双向绑定 注：其中可写任何可执行的Angularjs表达式 注：类似ng-bind

**ng-app** 声明anjularjs管理边界（定义一个ng应用程序）

**ng-init** 初始化ng应用程序变量，如ng-init="firstName='John'"

**ng-controller** 定义controller范围

**ng-model** 将元素值（如表单值）绑定到M

**ng-bind** 双向绑定 注：类似{{}}

**ng-clock** 全部加载前不显示内容 注：需指明calss=’ng-clock’

**ng-checked** 用于表单单选等是否被选中

**ng-selected** 用于表单多选、下拉等是否被选中

**ng-disabled** 用于表单验证，将其置为不可用，多用于按钮 注：其值为表达式，如，id.$invalid，需在要验证的表单项添加required属性

**ng-submit** 提交事件

**ng-src** 指定src 注：其值是{{}}

**ng-repeat** 循环 注：**$index**数组下标,**$first**是否为第一个,**$last**是否为最后一个

**css类**

直接定义表单的css中的这些类，对应状态会显示对应效果

**ng-empty**

**ng-not-empty**

**ng-touched**

**ng-untouched**

**ng-valid**

**ng-invalid**

**ng-dirty**

**ng-pending**

**ng-pristine**

**事件**

**ng-click** 点击事件 注：传递参数时可以将scope的属性直接传入

**ng-show** 是否显示 注：其值是表达式

**ng-hide** 是否隐藏 注：其值是表达式

**ng-if** 是否显示 注：其与ng-show的区别是它将移除dom，而ng-show和ng-hide是display:none

**ng-class** 是否应用css的class 注：其值是{‘class名’:表达式}

**动画**

AngularJS 使用动画需要引入 angular-animate.min.js 库。

还需在应用中依赖模块 ngAnimate

**ng-show**

**ng-hide**

**ng-if**

**ng-class**

**ng-view**

**ng-include**

**ng-repeat**

**ng-switch**

ng-show 和 ng-hide 指令用于添加或移除 ng-hide class 的值。

其他指令会在进入 DOM 会添加 ng-enter 类，移除 DOM 会添加 ng-leave 属性。

当 HTML 元素位置改变时，ng-repeat 指令同样可以添加 ng-move 类 。

此外， 在动画完成后，HTML 元素的类集合将被移除。例如： ng-hide 指令会添加一下类：

ng-animate

ng-hide-animate

ng-hide-add (如果元素将被隐藏)

ng-hide-remove (如果元素将显示)

ng-hide-add-active (如果元素将隐藏)

ng-hide-remove-active (如果元素将显示)

**module.directive(key,function(){return…})** 创建指令 注：返回对象所包含属性:restrict，E元素、A属性、Cclass；templage表示html代码（E）；replace是否替换自定义标签（E）；link执行的函数(A、C、E) ，link函数参数，scope控制器的scope/element所在元素/attrs所有属性/controller；controller控制器函数（E）；scope元素的scope（E）；require需要的元素（A），此时link函数有第四个参数代表改元素的controller

**过滤器**

过滤器可以使用一个管道字符（|）添加到表达式和指令中。

**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。

**angular.module(‘模块名’[,其他模块数组])** 定义模块 注：第二个参数是依赖的模块

**控制器**

**module.controller(‘控制器’,function($scope[,$rootScopt,$location,$http等]){})** 定义控制器 注：**$scope**可以看作mv 注：$scope见作用域

**控制器别名** 格式ng-controller=”控制器名 as 别名” 别名在控制器配置函数中使用this定义属性方法 在angular表达式中使用“别名.xxx”的方式调用别名中的属性方法 别名与控制其自身的方法不能混用，即不能用如上方式调用$scope定义的属性方法

**作用域**

$scope与$rootScopt。$scopt是本控制器内的作用域。而$rootScopt是全局作用域，它绑定在ng-app上，可以在本控制器外访问。

$scope实际上是创建控制器时，系统使用$injector创建的

除ng-controller外，ng-repeat指令也会创建自己的作用域

**$scope.$parent** 指向scope的父作用域；

**$scope.$$childHead** 指向scope的第一个子作用域；

**$scope.$$childTail** 指向scope的最后一个子作用域；

**$scope.$$nextSibling** 指向scope的下一个相邻作用域；

**$scope.$$prevSibling** 指向scope的上一个相邻作用域；

**$scope.$watch(watchExpression, listener, objectEquality)** 监听变化 注：第一个参数是监听的对象，可以是ng表达式或函数；第二个参数是回调函数，它有三个参数，newValue,oldValue,scope；第三个参数是深度监听，设置为true可监听对象的属性

**$scope.$descdory()** 清理作用域

**控制器事件**

**$scope.$broadcast(事件名,事件内容str)**  向本控制器和子控制器发送事件

**$scope.$emit(事件名,事件内容str)** 向本控制器和父控制器发送事件

**$scope.$on(事件名,function(event,data))**  接收事件 回调函数拥有的属性方法如下：

event.targetScope，发出或者传播原始事件的作用域；

event.currentScope，目前正在处理的事件的作用域；

event.name，事件名称；

event.stopPropagation()，一个防止事件进一步传播(冒泡/捕获)的函数(这只适用于使用`$emit`发出的事件)；

event.preventDefault()，这个方法实际上不会做什么事，但是会设置`defaultPrevented`为true。直到事件监听器的实现者采取行动之前它才会检查`defaultPrevented`的值；

event.defaultPrevented，如果调用了`preventDefault`则为true

**服务**

见module.controller

**$location** 用于获取页面的url

**$location.absUrl()** 得到页面的url

**$http** 用于发送请求

**$http.get(地址)** get请求 注：$http返回一个promise对象，其success的回调方法中有一返回数据参数，其error回调方法中有一参数是错误信息

**$http.post(地址,[参数])** post请求 注：参数是一个对象

**$http.delete(地址)** delete请求

**$http.put(地址,[参数])** put请求 注：参数是一个对象

**$timeout(function,time)** 对应于js的setTimeout

**创建服务与依赖注入**

**app.vlue(key,value)** 创建值

**app.module.constant(key,value)** 创建常量

**app.module.factory(key,function([依赖的其他服务]){return…})** 工厂模式创建对象 注：创建对象时可以将其他服务作为依赖注入

**app.module.service(key,function(){this….=…})** 服务模式创建对象 注：相当于使用new创建 注：可以利用其共享数据

**promise**

**promise.success(回调函数)**

**promise.error(回调函数)**

**promise.then(回调函数)**

**路由**

需要加载angular-route.js，app依赖ngRoute模块，路由设置通常在config方法中进行

路由是应用于整个app的

**ng-view** 使用该指令div内容随路由变化

**$routeProvider.when(paht,object)** 第一个参数是url；第二个参数是路由配置对象

**$routeProvider.otherwise(object)**

**路由配置对象**

$routeProvider.when(url, {

template: string, //简单html内容

templateUrl: string, //模版文件路径

controller: string, function 或 array, //在当前模板上执行的controller名，生成新的scope

controllerAs: string, //为controller指定别名

redirectTo: string, function, //重定向的地址

resolve: object<key, function> //注入当前controller所依赖的其他模块

});

**DOM操作**

dom操作最好在自定义的指令中进行

**angular-ui-router**

angular-ui-router是一个angular路由组件，它比angular本身提供的angular-router更加强大。路由的核心是给应用定义“状态”

**module.config(function($stateProvider,$urlRouteProvider){})**

**$urlRouteProvider.when(“路由url”,”路由url”);** 路由重定向

**$urlRouteProvier.otherwise(“路由url”)** 如果没有路由引擎匹配，则默认匹配到该url

**$stateProvider.state(“状态名”，路由配置对象)** 状态配置 注：状态名格式：父状态.子状态 路由配置对象如下：

url，路由url

template，简单的html

templateUrl，模版路径

controller，模版控制器。可以直接写控制器名；或者是一个数组，如['$scope', '$state', 'contacts', 'utils',function ($scope, $state, contacts, utils)]，最后一个函数是控制器配置函数

controllerAs，控制器别名

views，view 用在该状态下有多个 ui-view 的情况，可以对不同的 ui-view 使用特定的 template, controller, resolve data；绝对 view 使用 '@' 符号来区别，比如 'foo@bar' 表明名为 'foo' 的 ui-view 使用了 'bar' 状态的模板(template)，相对 view 则无

abstract，抽象state，抽象后该状态不能被链接，如果链接将链接到默认路由

**ui-sref** 链接到特定状态

**angular-ui-bootstrap**

ui-serf传递参数时，必须先在url中声明

**angular-loading-bar**

angular-loading-bar是一款加载进度条插件，在使用$http服务读取数据时自动显示进度条