（一）ng-app

①在标签里有ng-app属性时，表示声明这个标签里（包含他的所有子结点）的所有内容，都属于这个AngularJS应用。

想让代码生效，可以加载以下这些地方：

<!DOCTYPE html>  
<**html**> <!--ng-app可以加在这里-->  
<**head**>  
 <**title**>test</**title**>  
 <**script** src="angular-1.5.8/angular.js"></**script**>  
</**head**>  
<**body**> <!--ng-app或加在这里-->  
<**div** ng-app> <!--ng-app也可以加在这里-->  
 <**input** ng-model="name" type="text" placeholder="Your name">  
 <**h1**>Hello {{name}}</**h1**> <!--ng-app但不能只加在这里-->  
</**div**>  
</**body**>  
</**html**>

但注意，不能只加在h1中，或者在h1和input位置都加这个属性，这是无效的

②注意，以下这种是**不可以**的：

<**div** ng-app>  
 <**input** ng-model="name" type="text" placeholder="Your name">  
</**div**>  
<**div** ng-app>  
 <**h1**>Hello {{name}}</**h1**>  
</**div**>

如果想让input和这里的name互动，必须放在同一个带ng-app的dom下。

③变种一：

<**input** ng-model="name" type="text" placeholder="Your name">  
<**h1**>Hello {{name}}</**h1**>  
<**input** ng-model="name" type="text" placeholder="Your name">

以上这种写法，无论更改哪个输入框，都会导致另外一个输入框和<h1>标签中的值跟着一起改变；

④变种二：

<**input** ng-model="name" type="text" placeholder="Your name">  
<**input** value="{{name}}" type="text" placeholder="Your name">

以上这种写法，当未主动修改第二个输入框时，第一个输入框的值的改变，会导致第二个输入框的值跟着一起改变；

但假如改变了第二个输入框的值，那么第一个输入框的值将不再link第二个输入框了。

即使你重新让两个输入框的值一样，也不可以。（推测是因为{{name}}的值被改变，导致无法link在一起）

⑤假如添加一个输出输入框值的按钮的事件：

<**body**>  
<**script**>  
 function test() {  
 var m = **document**.getElementById('a').**value**;  
 **console**.log(m)  
 }  
</**script**>  
<**div** ng-app>  
 <**input** ng-model="name" type="text" placeholder="Your name">  
 <**input** ng-model="name" type="text" id='a' placeholder="Your name">  
 <**input** type="button" onclick="test()">  
</**div**>  
</**body**>

无论是③或者④中的写法，或者是④中的两种情况，都可以保证正常输出第二个input中的值。

**（二）将controller、module绑定起来**

（1）在老版本（angular1.3之前）里，是这么绑定的：

<**body** ng-app>  
<**div** ng-controller="MyController">  
 <**h1**>Hello {{ clock }}!</**h1**>  
</**div**>  
<**script** type="text/javascript">  
 function MyController($scope, $timeout) {  
 var updateClock = function() {  
 $scope.**clock** = new Date();  
 $timeout(function() {  
 updateClock();  
 }, 1000);  
 };  
 updateClock();  
 };  
</**script**>  
</**body**>

将ng-app放在根标签里，然后ng-controller和函数名相同，从而进行操作。

假如提示xxxxx is not a function，而这里的xxxx是ng-controller里面的名字的话，那么可能是因为你的angularJs的版本较新，需要看下面的内容。

（2）新版本

然而，因为在angularJs1.3版本中，为了让根节点不再被挂载很多亢余的内容，所以禁止直接在根节点上注册controller。因此，需要通过另外的方式来将controller注册到对应的模型上：

<!DOCTYPE html>  
<**html** ng-app="MyApp"> <!--ng-app可以加在这里-->  
<**head**>  
 <**title**>test</**title**>  
 <**script** src="angular-1.5.8/angular.js"></**script**>  
  
</**head**>  
<**body** ng-controller="MyController">  
<**script**>  
 angular.module('MyApp', []).controller('MyController', function ($scope, $timeout) {  
 var updateClock = function () {  
 $scope.**clock** = new Date();  
 $timeout(function () {  
 updateClock();  
 }, 1000);  
 }  
 updateClock();  
 })  
</**script**>  
<**ul**>  
 当前时间：{{clock}}  
</**ul**>  
</**body**>  
</**html**>

例如在这段代码里，就是指将控制器绑定在MyApp这个模块下。

效果是先将clock这个变量用new Date()赋值，然后启用定时模块，每秒更新一次时间。

（3）setInterval代码里，$scope代码的值将被更新，但不会反应在页面上。只要向上面那样使用$timeout才可以更新

按照我的理解，之所以会这样，是因为脏值检测的原因，单纯靠setInterval是不会触发脏值检测的，那个监测是应该是DOM

**（三）js和html分离**

（1）分离前代码：

<!DOCTYPE html>  
<**html** ng-app="MyApp"> <!--ng-app可以加在这里-->  
<**head**>  
 <**title**>test</**title**>  
 <**script** src="angular-1.5.8/angular.js"></**script**>  
  
</**head**>  
<**body** ng-controller="MyController">  
<**script**>  
angular.module('MyApp', []).controller('MyController', function ($scope, $timeout) {  
 var updateClock = function () {  
 $scope.**clock** = new Date();  
 $timeout(function () {  
 updateClock()  
 }, 1000)  
 }  
 updateClock();  
})

</**script**>  
<**ul**>  
 当前时间：{{name[0]}}  
</**ul**>  
</**body**>  
</**html**>

（2）分离后

先在同目录下创建**test.js**文件，当前的html文件应为**test.html**。

test.html

<!DOCTYPE html>  
<**html** ng-app="MyApp">  
<**head**>  
 <**title**>test</**title**>  
 <**script** src="angular-1.5.8/angular.js"></**script**>  
  
</**head**>  
<**body** ng-controller="MyController">  
<**script** src="./test.js"></**script**>  
<**ul**>  
 当前时间：{{clock}}  
</**ul**>  
</**body**>  
</**html**>

test.js文件：

angular.module('MyApp', []).**controller**('MyController', function ($scope, $timeout) {  
 var updateClock = function () {  
 $scope.**clock** = new Date();  
 $timeout(function () {  
 updateClock()  
 }, 1000)  
 }  
 updateClock();  
})

（3）html和js的分离是十分重要的，可以方便针对性维护代码，也方便了视图控制器分离；

**（四）绑定**

（1）只更新部分的值，将更新函数绑定起来

但具体还不是很清楚，推测$apply的作用类似于监视比如输入框的$watch，会在执行参数中函数的时候，更新视图模板中的内容

angular.module('MyApp', []).controller('MyController', function ($scope, $timeout) {  
 $scope.**clock** = {  
 **now**: new Date()  
 }  
 var updateClock = function () {  
 $scope.**clock**.**now** = new Date();  
 }  
 setInterval(function () { //设置延迟执行函数  
 $scope.$apply(updateClock); //这个大概是将updateClock的函数执行和$scope绑定起来，使其能更新scope的值  
 }, 100);  
 updateClock();  
})

**（五）angularJs的模块的声明和引用**

（1）声明模块：

angular.module('MyApp', [])

①第一个参数是模块的名称；

②第二个参数是模块的依赖列表，是一个数组；依赖会在声明当前模块之前加载。

③按照说明，相当于AngularJS模块的setter方法，用于定义模块的。（但我不太清楚setter方法是什么）

（2）引用模块：

angular.module('MyApp')

①参数是模块的名称；

②相当于AngularJS里的getter方法（但我还是不懂）

③然后可以通过在上面代码的返回的对象中，创建应用了。

（3）示例：（以下不能正常运行！！只是备用！！）

<!DOCTYPE html>  
<**html**>  
<**head**>  
 <**title**>test</**title**>  
 <**script** src="angular-1.5.8/angular.js"></**script**>  
  
</**head**>  
<**body**>  
<**script** src="./test.js"></**script**>  
<**div** ng-app="MyApp">  
 <**ul** ng-controller="MyController">  
 当前时间：{{clock.now}}  
 </**ul**>  
</**div**>  
  
<**div** ng-app="AnotherApp">  
 <**h1** ng-controller="AnotherController">  
 每三秒同步一次的时钟: {{anotherClock}}  
 </**h1**>  
</**div**>  
</**body**>  
</**html**>

angular.module('MyApp', []).controller('MyController', function ($scope, $timeout) {  
 $scope.**clock** = {  
 **now**: new Date()  
 }  
 var updateClock = function () {  
 $scope.**clock**.**now** = new Date();  
 }  
 setInterval(function () { //设置延迟执行函数  
 $scope.$apply(updateClock); //这个大概是将updateClock的函数执行和$scope绑定起来，使其能更新scope的值  
 }, 100);  
 updateClock();  
})  
  
angular.module('AnotherApp', ['MyApp']).controller('AnotherController', function ($scope, $timeout) {  
 $scope.**anotherClock** = new Date();  
 //$scope.anotherClock = $scope.clock.now;  
 var Synchronize = function () { //每三秒同步一次  
 $timeout(function () {  
 $scope.**anotherClock** = $scope.**clock**.**now**;  
 Synchronize();  
 }, 3000)  
 }  
  
 Synchronize();  
})

**（六）作用域**

①作用域是表达式执行的上下文；

②$scope对象是定义应用逻辑、控制器方法和视图属性的对象。虽然他是回调函数的第一个参数，但他 **不能被更名** ，例如不能改为$SCOPE或者其他的。

——关键就是必须用$scope这个变量名

③在应用将视图渲染并呈现给用户之前，视图中的模板会和作用域进行连接，然后应用会对DOM进行设置，以便将属性变化通知给AngularJS。

——按我理解，就是将ng-app和对应的模块连接起来，让两个其中任何一个变化，都会影响另外一个

④$scope是树形结构，可以多层嵌套

当参数是 $rootScope 时，会将ng-app和$rootScope进行绑定。他是所有 $rootScope对象的最上层。

angular.module('MyApp', []).controller('MyController', function ($scope, $rootScope) {  
 **console**.log($scope);  
 **console**.log($rootScope);  
})

返回值依次为：

$scope的值

{  
 **$$ChildScope**: null,  
 **$$childHead**: null,  
 **$$childTail**: null,  
 **$$listenerCount**: Object,  
 **$$listeners**: Object,  
 **$$nextSibling**: null,  
 **$$prevSibling**: null,  
 **$$watchers**: Array[1],  
 **$$watchersCount**: 1,  
 **$id**: 2,  
 **$parent**: Scope,  
 **\_\_proto\_\_**: Scope  
}

$rootScope的值

{  
 **$$ChildScope**: ChildScope(),  
 **$$applyAsyncQueue**: Array[0],  
 **$$asyncQueue**: Array[0],  
 **$$childHead**: ChildScope,  
 **$$childTail**: ChildScope,  
 **$$destroyed**: false,  
 **$$isolateBindings**: null,  
 **$$listenerCount**: Object,  
 **$$listeners**: Object,  
 **$$nextSibling**: null,  
 **$$phase**: null,  
 **$$postDigestQueue**: Array[0],  
 **$$prevSibling**: null,  
 **$$watchers**: null,  
 **$$watchersCount**: 1,  
 **$id**: 1,  
 **$parent**: null,  
 **$root**: **Scope**,  
 **\_\_proto\_\_**: Object  
}

（2）$scope

①$scope.$parent可以找到$rootScope

②$rootScope.$$childTail或者是$rootScope.$$childHead都可以找到$scope；

但之所以这样，是因为只有这一个子，假如有多个子的话，那么分别找到最后一个$scope或者第一个$scope了

③$scope对象是页面和代码之间的桥梁，可以在代码中通过它来操纵视图中的变量；

他的所有属性，都可以在视图中被访问到；

但并不意味着他的值的更新，就会让页面中的对应值更新。

典型的来说，如以下代码：

**angular**.module('MyApp', []).controller('MyController', function ($scope) {  
 $scope.**clock** = {};  
 $scope.**clock**.**now** = new Date();  
 var update = function () {  
 $scope.**clock**.**now** = new Date();  
 };  
 update();  
 setInterval(update, 1000);  
 setTimeout(function(){  
 **console**.log($scope.**clock**.**now**)  
 },3000);  
})

页面里显示的内容为：

当前时间："2016-08-11T12:23:44.236Z"

而实际变量的值在3秒后是：

Thu Aug 11 2016 20:23:47 GMT+0800 (中国标准时间)

这说明了两件事：

《1》日期在页面上显示的时候是被处理过的；

《2》单纯的改变变量值，并不影响页面上的值，除非像之前那样用$scope.$apply()来绑定某个函数，使得其调用的时候可以影响；

除此之外，$scope的变量可以是一个函数，在页面中调用时，就像调用函数那样，例如代码是：$scope.nowTime = function(){ return xxxx}; 页面中是{{nowTime()}}

——————以下纯理论不太理解，等结合代码再回顾——————

（3）AngularJS的模板

个人理解，模板范围内就是作用域，模板就是作用域所包含的范围

①可以使用多种标记：

《1》指令：将DOM元素增强为可复用的DOM组件的属性或元素；（比如之前的ng-model=”xx”？）

《2》值绑定：模板语法{{}}可以将表达式绑定在视图上；（之前的{{clock}}）

《3》过滤器：可以在视图中使用的函数，用来进行格式化；

《4》表单控件：用来检验用户输入的控件；

②功能：

《1》提供观察者以监视数据模型的变化；

《2》可以将数据模型的变化通知给整个应用，甚至是系统外的组件；

《3》可以进行嵌套，隔离业务功能和数据；

《4》给表达式提供运算时所需的执行环境。

（4）$scope的声明周期

①创建：创建控制器或者指令时

②链接：Angular开始运行时，所有的$scope对象被附加或者链接到视图中，所有创建$scope对象的函数也会将自身附加到视图中。他们会注册当Angular应用上下文中发生变化时需要运行的函数。

他们是$watch函数，AngularJS通过这些函数获知何时启动事件循环。

③更新：事件循环运行时，被执行在顶层的$scope对象（即$rootScope）；

每个子作用域都执行自己的脏值检测，每个函数都会检查变化，检测到则触发指定的回调函数；

④销毁：当一个$scope不需要时，这个作用域会自动被清理；

如果想自己销毁，那么就调用$scope的$destory()方法来清理

（5）指令和作用域

指令被广泛应用，指令通常不会创建自己的$scope，但有例外，比如ng-controller和ng-repeat指令会创建自己的子作用域并将它们附加到DOM元素上。

**（七）控制器**

（1）AngularJS的控制器是一个函数，用来向视图中的作用域添加额外的功能。

例如：

angular.module('MyApp', []).controller('MyController', function ($scope) {  
 $scope.**clock** = {};  
 $scope.**clock**.**now** = new Date();  
})

这里的controller中的第二个参数是一个函数，而这个函数就是控制器。

注：

在之前版本中，控制器可以是一个和controller中第一个参数同名的函数（只不过这样的控制器是一个全局的控制器，并不好）

可以将代码分拆为两部分：

var MyApp = angular.module('MyApp', []);  
MyApp.controller('MyController', function ($scope) {  
 $scope.**clock** = {};  
 $scope.**clock**.**now** = new Date();  
})

这是创建一个模块，然后为模块创建了一个控制器

（2）ng-click

①在DOM的标签上使用ng-click，可以将点击事件和DOM元素进行绑定；

②具体如何处理，在控制器里进行定义；

③该事件指浏览器的mouseup事件；

示例：

<**div** ng-app="MyApp">  
 <**ul** ng-controller="MyController" ng-click="update()">  
 当前时间：{{clock.now}}  
 </**ul**>  
</**div**>

var MyApp = angular.module('MyApp', []);  
MyApp.controller('MyController', function ($scope) {  
 $scope.**clock** = {};  
 $scope.update = function(){  
 $scope.**clock**.**now** = new Date();  
 }  
 $scope.**clock**.**now** = new Date();  
})

效果是点击该DOM后，会让时间更新为最新的时间。

在DOM标签里，调用的是update()这个函数，并且，他是可以传参数的。

（3）标签里的this：

例如：

ng-click="update(**this**)"

标签里的this是$scope这个对象

（4）控制器的父级作用域：

①AngularJS应用的任何一部分，无论渲染在哪个上下文，都有父级作用域存在；

②对子级$scope来找父级作用域的方法是：$scope.$parent;

示例：

html

<**div** ng-app="MyApp">  
 <**div** ng-controller="ParentController">  
 <**ul** ng-controller="MyController" ng-click="update(**this**)">  
 当前时间：{{clock.now}}  
 </**ul**>  
 </**div**>  
</**div**>

Js

var MyApp = angular.module('MyApp', []);  
MyApp.controller('ParentController', function ($scope) {  
 **console**.log($scope); //这里是父级的，id为2  
})  
MyApp.controller('MyController', function ($scope) {  
 **console**.log($scope.**$parent**); //单纯的$scope是子级的，id为3；但加上$parent后，就找到了id为2的$scope了  
})

**③问题：**

但目前不知道如何从父级$scope来获取子级$scope；难道用$scope.$parent.MyController = $scope这种方法么？

理论上也可以，但需要考虑到先执行父级$scope再执行子级的问题，简单来说，就是在进行这样的赋值前（执行这部分代码前），父级$scope是无法通过这样来获取子级$scope的。

示例：（可以显示）

var MyApp = angular.module('MyApp', []);  
MyApp.controller('ParentController', function ($scope) {  
 setTimeout(function(){  
 **console**.log($scope.**MyController**);

},1000)  
})  
MyApp.controller('MyController', function ($scope) {  
 $scope.**$parent**.**MyController** = $scope;  
 **console**.log($scope.**$parent**);

})

示例：（undefined）

var MyApp = angular.module('MyApp', []);  
MyApp.controller('ParentController', function ($scope) {  
 **console**.log($scope.**MyController**);  
})  
MyApp.controller('MyController', function ($scope) {  
 $scope.**$parent**.**MyController** = $scope;  
 **console**.log($scope.**$parent**);  
})

**（八）表达式**

（1）特性：

①表达式在其所属作用域执行，有访问本地$scope的权限；

②发生TypeError和ReferenceError并不会抛出异常；

③不允许使用任何流程控制功能（条件控制，例如if/else（书上），我推测while之类也不可以）

④可以接受过滤器和过滤器链**（不懂）**

**（2）$parse 表达式解析**

简单来说，这个用于解析表达式。

例如，在input文本输入框里有一个12+3的内容，在一般情况下，数据绑定后，会认为他是一个字符串；

但是假如调用这个$parse来解释，那么会认为其是一个表达式，那么返回值就是15

示例：

html：（模块略）

<**div** ng-controller="MyController">  
 <**input** ng-model="input" type="text"/>  
 当前时间：{{text}}  
</**div**>

js代码：（模块略）

MyApp.controller('MyController', function ($scope, $parse) {  
 $scope.$watch("input", function (newVal, oldVal, scope) {  
 $scope.**text** = $parse(newVal)();  
 })  
})

效果：

①在输入普通数字时，会正常显示，例如1；

②在输入字母之类时，无反应，例如a；

③输入运算符时，会提示一次错误，例如 12+ 或 +

④输入完整的正确表达式时，右边输出正确结果，例如12+3 显示15。

另外，$parse()的第一个对()的参数表示他要解释哪个表达式，而第二对()表示他要执行。

第二对括号里可能能添加参数（不确定，推测是作用域）

**（3）$interpolate 插值字符串**

（1）简单来说，他的功能有：

①定义符号的开始和结束（例如把{{name}}才能被转换，变为--name--这样格式）；

②