自定义指令入门

指令是模板上打给AngularJS的标记。
它可能是一个自定义的标签名或者一个自定义的属性名。（或者是CSS类或者注释）

AngularJS中，自定义指令的API被设计在模块(module)对象上，和Controller与Service同级，是AngularJS功能组件之一。

来看一下AngularJS的指令是如何声明的：

```
var app = angular.module('demo.main',['demo.directives']);

var directives = angular.module('demo.directives',[]);

directives.directive('demoHello',function(){
    return {
        restrict:"EA",
        template:"<span>hello world</span>"
    }
});
```

通常来说，AngularJS应用的自定义指令可以放入一个单独的模块中，统一管理。

定义自定义指令的名字时，记得必须符合驼峰命名法，因为在用指令打标记时，AngularJS会把驼峰命名法的指令拆成对应的格式，如demoHello指令在作为标签名和属性名时有如下用法：
<demo-hello></demo-hello>
<div demo-hello>

自定义指令时回调函数返回的对象用于描述自定义指令的各种功能。
在这里，restrict表示指令生效的方式，E代表这个指令可以作为标签名生效，A代表这个指令可以作为属性名生效，EA则代表可以同时作为标签名和属性名生效。实际上还有作为CSS类生效和作为注释生效的C和M，但是不建议使用。

AngularJS自定义指令的使用方法：

```
<div demo-hello="">this is a div</div>
<demo-hello>this is a div</demo-hello>
```

自定义指令：模板

AngularJS自定义指令的描述对象中，有这些字段可以进行设置：

restrict：约束，指定指令生效的方式。可以是字符串E、A或者EA。
template/templateUrl：指令的模板字符串或者指令的模板文件地址，AngularJS会用它们的内容替换指令指定的标签的内部的内容。
transclude：内嵌，true/false，当为true时，指令内部模板用ng-transclude指定一个标签（假设它是ElemInner），指令本身所指定的标签（假设它是ElemOuter）内部的全部内容会被放到ElemInner里面去。
replace：替换，true/false，当为true时，AngularJS用模板的内容替换指定标签本身。（这需要模板内部所有标签都被一个容器标签包裹起来）

自定义指令：作用域

AngularJS的自定义指令可以自带一个作用域。

在描述对象中，用scope属性可以设置指令内部作用域和外部作用域绑定的方式，如：

```
scope:{
        "oneWay" : " < ",
        "twoWay" : " = ",
        "string" : " @ "
}
```

在使用这个指令时，就可以通过属性的方式设置字符串或者外部作用域的值和表达式与指令内部作用域互相绑定，如下：

```
<demo-directive
        one-way="data.getNum()"
        two-way="data.name"
        string="直接输入字符串就可以了" ></demo-directive>
```

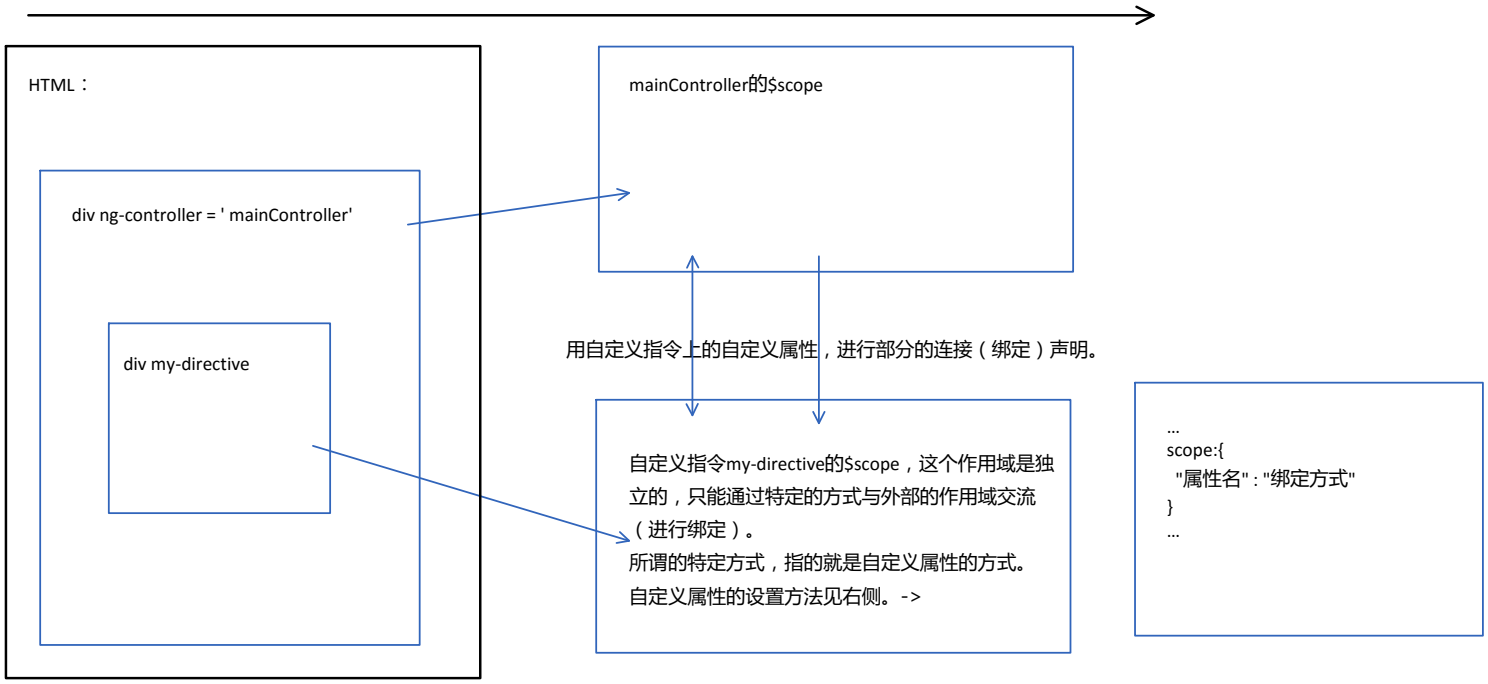
注意如下细节：

scope对象中使用的格式是：

```
{
        属性名1：属性绑定方式,
        属性名2：属性绑定方式,
        ..............
}
```

属性绑定方式中：

< 代表单向绑定，使用时可以输入AngularJS表达式。指令内部作用域得到的是表达式执行后的值。
= 代表双向绑定，使用时可以输入外部作用域上的字段（属性）名。将会把指令内部作用域的某个字段和指令外部作用域的某个字段绑到一起，其中一个修改另外一个跟着修改。
@ 代表字符串绑定，使用时直接输入字符串，指令内部会把它的内容当做字符串绑定进去。注意：这个属性内部可以使用{{}}差值表达式。



自定义指令：控制器和链接函数

自定义指令中可以声明控制器：

```
controller:function($scope, $element, $attrs){
        // Todo: 各种数据操作

        return {
                ....
        }
}
```

指令的控制器也可以使用依赖注入写法，和普通的控制器是一样的。
在指令的控制器中可以操作指令的作用域，甚至是指令绑定的元素。
注意，$element是个jQLite对象，不是纯JavaScript的DOM对象。另外，如果引入了jQuery的话，$element就会是jQuery对象。

在自定义指令中可以声明链接函数：

```
link:function(scope,element,attrs,ctrl){
        // Todo: 各种元素操作
}
```

链接函数的参数表是定死的，不能写依赖注入，四个参数是有顺序的，依次是指令的控制器，指令绑定的元素，指令绑定元素的各个属性，和Controller的返回值（这个后面会讲）。
用法和Controller有些类似。

但是，指令的第四个参数Ctrl可以接受一个控制器的返回值。至于接受哪个控制器的返回值呢？我们可以用

```
require:"^outerDirective"
```

的方式来指定。^outerDirective代表在这个指令的外侧寻找一个名字叫outerDirective的指令，并把它的Controller的返回值作为链接函数的第四个参数传给链接函数

通常来说，AngularJS是一个不需要文档操作的框架。但是有时情况特殊，也要做一些文档操作。
这个时候，DOM操作可以放到指令的link函数里面去。至于指令的Controller函数，通常来说做一些数据处理，或者干脆直接使用link函数。
至于依赖注入，我们其实是可以直接在声明指令时，进行依赖注入。（因为曾经指令时，也使用了一个可以依赖注入的回调函数）