**彻底弄懂transclusion**

AngularJS中指令的重要性是不言而喻的，指令让我们可以创建自己的HTML标记，它将自定义元素变成了一个一个的模块，极大的体现了前端开发中的模块化模式，并提高了代码的易读性和重用性。AngularJS中的指令也是学习AngularJS中的一个难点所在，其中的许多属性，需要反复学习，认真体会，方能领悟其中的精妙之处。

今天我们要讲的就是其中一个重点和难点 – transclusion。关于这个话题我之前也写过很多文章来讲述，但是当时都是照搬博文中的例子，自己也没有比较深刻的体会，因此一直不得要领。今天我们的目标就是“彻底弄懂transclusion”。

**1.什么是transclusion**

好吧，我知道你肯定会去查词典，但是你会发现，词典上没有transclusion这个词的准确释义！！！纳尼！！！这不坑爹的吗！！！！

还好，维基百科上有一个注释，翻译过来意思大概是这样的：

transclusion在计算机科学中指的是讲一个文档或者一个文档的某部分在另一个文档中引用。

我去，这不坑爹的吗！！！什么意思！！！！

确实，你猜对了，这个解释对我们一点帮助都没有！！！！！！

还好，我们终于在某道词典找到了一个解释:“嵌入”，这里指的是transclusion这个词，而在后面我们即将看到的transclude这个词压根在词典上就找不到。算了，淡定一点，继续往下看。其实这里翻译为“嵌入”，如果从实际运用中来看，还是比较贴切的。如果你不太理解什么是transclusion，我们下面用一个例子来说明一下。

ok，现在我们要创建一个指令了，我们把这个指令叫做<handsome-me>。在此先略过这个指令的创建过程，如果你还不知道怎样创建一个指令，请前参看前面几篇文章。好了，无论怎么说，这个指令已经创建好了，于是我们可以有以下几种用法：

第一种：

<handsome-me/>

就像是input一样对吧，很简单，我们叫它“自开自闭”（要是你在别的书上没看过这个名称，那就是我发明的，反正就这么叫）标签。

第二种：

<handsome-me></handsome-me>

就像是div一样对吧，更简单，我们叫它“自开别人闭”（同上同上）标签。

第三种：

<handsome-me>

//中间有好多代码

</handsome-me>

这个和第二种很像吧，但是又有点区别，我们叫它“自开别人闭，中间加一坨”（总是感觉好粗俗。。。never mind。。。）标签。

我们来对比以下三种标签，它们有什么区别。当然区别很多。但是具体到我们今天的话题，与之相关的最大的区别就是前两种中间没有加一坨，第三种中间加了一坨。OK，因此我们现在来总结什么叫做transclusion：

**如果你在定义指令的时候，想要它在具体使用时中间加一坨，那么你就要用transclusion。**

这个定义实在是太经典了，完全比什么官方文档要清楚有没有，完爆各种老外唧唧歪歪说半天还是不明白有没有，完全符合中国国情有没有！！！

好了，知道了定义以后，我们要开始来看看具体怎么使用transclusion了。如果你了解AngularJS指令的编写，你一定知道return的那个对象的tranclude指令默认是false，因此如果你想要开启使用transclusion的话，就要将这个transclude属性赋上一个别的值，当然，这个值不能乱赋，它只有两种选择：

第一种选择：

transclude: true

第二种选择：

transclude: 'element'

我去，这个又是毛线啊！两者之间有毛的区别啊！！文档完全是看不懂的嘛！！！

淡定一点，现在我们来说区别。最常用的呢，是第一种，也就是赋值为true。还记得transclusion的中文意思吗，“嵌入”对吧！因此我们现在就不说“一坨”，而把中间的这一坨叫做“嵌入部分”。ok，回到正题，当transclude是true的时候，嵌入部分就是嵌入部分，比如说：

<handsome-me>

{{name}}

<handsome-me>

在transclude:true的时候，它的嵌入部分是什么啊？对了，就是{{name}}。再来一发：

<handsome-me>

<div>

<span>{{name}}</span>

</div>

<handsome-me>

在transclude:true的时候，它的嵌入部分是什么啊？对了，是

<div>

<span>{{name}}</span>

</div>

太简单了是吧！so easy！妈妈再也不用担心我的学习！！！

现在再来讲第二种情况，当transclude的值是element的时候，又是怎样一种情形。此时，嵌入部分变成了原来的嵌入部分加上外边的自定义标签，也就是整个元素。又听不懂了！！！fork fork fork!!!!!!淡定，我们再来举例子：

<handsome-me>

{{name}}

<handsome-me>

在transclude:’element’的时候它的嵌入部分是什么啊？对了，是：

<handsome-me>

{{name}}

<handsome-me>

再来一发：

<handsome-me>

<div>

<span>{{name}}</span>

</div>

<handsome-me>

在transclude:’element’的时候它的嵌入部分是什么啊？对了，是：

<handsome-me>

<div>

<span>{{name}}</span>

</div>

<handsome-me>

都说的这么详细了，不要再说你不会了哈！！！

**2.ng-transclude的作用是什么**

在编写指令时，我们都会有一个template或者templateUrl这样的属性是吧。在使用transclusion时，我们要把嵌入部分放到模板中，因此我们有两种选择，其中一种选择就是使用ng-transclude。

ng-transclude是干什么用的，我们还是先来看定义，再来看例子：

**ng-tranclude决定了在什么地方放置嵌入部分。**

太好理解了！于是我们来看例子：

假设指令是这样的：

<handsome-me>

{{name}}

</handsome-me>

而模板是这样的：

<div>

<p>MaMa does not need to worry about my study anymore! </p>

<div ng-transclude></div>

</div>

于是，在transclude:true的情况下，最终呈现在页面中的HTML会是什么样子。对了，是这样：

<div>

<p>MaMa does not need to worry about my study anymore! </p>

{{name}}

</div>

另一种情况，在transclude:’element’的情况下，最终呈现在页面中的HTML会是什么样子。对了，是这样：

<div>

<p>MaMa does not need to worry about my study anymore! </p>

<handsome-me>

{{name}}

</handsome-me>

</div>

例子这么清楚，总能明白了吧！！

**3.不使用ng-transclude的情形**

OK，现在我们来想一个问题，如果我想把我的嵌入部分多次放入我的模板中怎么办？你可能会说，那就多放几个ng-transclude呗！这当然是不行的，在AngularJS中你只在一个指令的模板中只能申明一个ng-tranclude。所以这种情况下我们就能使用模板了，因此我们要使用一个叫做tranclude()的函数！！

纳尼！这又是什么东西！！！如果你仔细去研究一下AngularJS的文档的话，你一定会发现一个叫做$tranclude的service，它就是我们现在要将讲的东西。那么这个函数怎么用？如果你看过一些关于ng-repeat，ng-swift源码的解析，你一定会记得其中的一个叫做linker的东西。这个东西上是什么曾经困扰过我好长时间，但是后来我发现这个linker()其实就是transclude()。

我们在link,compile以及controller中都能找到这个transclude函数的身影。在link函数中，transclude是link函数的第五个参数；在compile函数中，transclude是compile函数的第三个参数。在这个两个函数中，由于我们没有使用依赖注入，因此只要顺序对了就对了，随便命名为什么都可以。而在controller函数中，由于使用的是依赖注入，因此transclude是$transclude，只要名字写对了就对了。在link，compile和controller函数中，transclude的用法一模一样，因此在这我们只举一个link函数的例子：

1.最简单的用法：

link(scope,elem,attrs,ctrl,transclude){

var content = transclude();

elem.append(content);

}

在这里，我们通过transclude()返回了嵌入部分的具体内容，然后append到了元素的elem的尾巴上，当然，你想要append多次也是可以的。

2.复杂一点的用法：

link(scope,elem,attrs,ctrl,transclude){

tranclude(scope,function(clone){

elem.append(clone);

})

}

这里tranclude接受了两个参数，第一个是scope，代表作用域。第二个回调函数中带有一个参数clone，其实它就是嵌入内容，和transclude()的返回值一模一样。那么前面的第一个参数的scope有什么用呢？这就要说到transclude和作用域了！

**4.transclude和scope**

我们知道，在定义一个指令时，如果不显式声明scope，那么指令的作用域就是父作用域。如果声明scope:true或者scope:{}，那么指令会生成一个自己的作用域，只不过一个原型继承，一个独立而已。如果你使用transclusion，那么无论什么情绪，都会生成一个新的作用域，这个作用域直接原型继承于父作用域，它的地位和指令生成的作用域是一样的，二者属于并列的关系。

于是我们现在就能了解tranclude(scope,function(clone){})中的scope是什么意思了，默认情况下，如果我们简单使用translude()，那么作用域默认的是transclude生成的自作用域。但是如果我们使用tranclude(scope,function(clone){})，那么作用域显然就是directive的作用域了。要是我们想使用父作用域怎么办，很简单：

tranclude(scope.$parent,function(clone){})

要是想要一个新的作用域怎么办，也很简单：

tranclude(scope.$parent.$new(),function(clone){})

你要是对作用域是什么东西，作用域是怎么继承的，那不是今天我们要讲的话题。

说了这么多，这么直白，想必你已经对AngularJS的transclusion彻底的清楚明白了吧。要是不明白，再看几遍，总会明白的！！！