**SOURCE:** **http://angular-tips.com/blog/2014/03/transclusion-and-scopes/**

**Transclusion and Scopes** Mar 8th, 2014

**The problem**

There is a common misconception that I see when I am doing Angular support. My goal here is to address it.

Let’s imagine I have this simple controller and directive:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | app.controller('MainCtrl', function($scope) {  $scope.person = {  name: 'John Doe',  profession: 'Fake name'  };  $scope.header = 'Person';  });  app.directive('person', function() {  return {  restrict: 'EA',  scope: {  header: '='  },  transclude:true,  template: '<div ng-transclude></div>',  link: function(scope, element, attrs) {  scope.person = {  name: 'Directive Joe',  profession: 'Scope guy'  };  scope.header = 'Directive\'s header';  }  };  }); |

I have a controller with a person and a header on the scope and I also have a person directive which also creates a person and modifies the header. The directive has an isolated scope, so it is not aware of the controller’s person. Let’s use it:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <body ng-controller="MainCtrl">  <person header="header">  <h2>{{header}}</h2>  <p>Hello, I am {{person.name}} and,</p>  <p>I am a {{person.profession}}</p>  </person>  </body> |

What is supposed to happen here? What should we see here? Let me think about it… We have a person directive which have a person on it called Directive Joe and also a header which says Directive's header. Then in our HTML we used the directive passing the controller’s header and then we put some HTML **inside** the directive. Alright, we should see the Directive's header and also the information about Directive Joe. That is obvious since the HTML inside the directive (which is called Transcluded html) is going to be transcluded into our directive. So our scopes are more or less like:

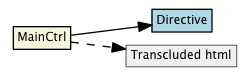
http://angular-tips.com/images/transclusionscopes/diag1.jpg

(The normal arrow is for new isolated scopes and the dashed is for new non-isolated scopes)

**Try it**

Wait a second, we have mixed result here… On the one hand, we have our Directive's header as expected but on the other hand, we got the controller’s John Doe person. That makes no sense at all. What’s going on?

The misconception here is to think that the transcluded html has access to the isolated scope or that the transcluded html is a new child scope of it (as I showed on the diagram before). The reality is that the transcluded html is a new child scope of the **controller’s** one. Yes, that is right:



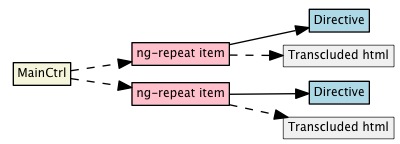
(The normal arrow is for new isolated scopes and the dashed is for new non-isolated scopes)

Having this in mind, the result makes more sense. The transcluded html only sees what is on the controller’s scope. For the person it is clear, it is showing it as is. But what about the header? It is showing the directive’s one. Well, that isn’t true. Since we created a two-way databinding on the header, when we changed the header on the directive, the controller’s one also changed. That is why we saw Directive's header, because the controller’s header was also updated.

So, the transcluded html is a new child scope of the current scope on that DOM. In this case, the controller's scope. In the case that you put an ng-repeat like:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <body ng-controller="MainCtrl">  <div ng-repeat="foo in foos">  <person header="header">  <h2></h2>  <p>Hello, I am and,</p>  <p>I am a </p>  </person>  </div>  </body> |

The scopes would be like:



(The normal arrow is for new isolated scopes and the dashed is for new non-isolated scopes)

**The ways around**

This is how the transclusion and its scope works by default. That doesn’t mean that we can’t do something to modify this behavior.

If we check the [documentation](http://docs.angularjs.org/api/ng/service/$compile) we can see that the link function of a directive is like:

|  |  |
| --- | --- |
| 1 | function link(scope, iElement, iAttrs, controller, transcludeFn) { ... } |

Uh, that fifth parameter says something about transclusion. With that function, we have control of both the scope and the HTML of the transclusion. Let’s see it:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | app.directive('person', function() {  return {  restrict: 'EA',  scope: {  header: '='  },  transclude:true,  link: function(scope, element, attrs, ctrl, transclude) {  scope.person = {  name: 'Directive Joe',  profession: 'Scope guy'  };  scope.header = 'Directive\'s header';  transclude(scope.$parent, function(clone, scope) {  element.append(clone);  });  }  };  }); |

**NOTE**: Link parameters are fixed parameters so doesn’t matter the name you give to them.

the transclude function receives a function and an optional first parameter. What this function does is to clone the transcluded html and then you can do with it what you want. If you put a scope as the first parameter, that scope will be the one used on the cloned element. The callback function of transclude will receive the cloned DOM and also the scope attached to it.

In this case, we are using the directive’s parent scope (in this case the controller’s one) as the scope of the transcluded html and then we are receiving it in the callback function. What we do here is just append it on our directive’s DOM element. In the case we had a template on the directive, we could retrieve a DOM element and then use it to append the transcluded html, that is what I call complete control :)

**Try it**

The result is the same visually, but internally the transclusion did not create a new scope, it is using the controller’s one.

http://angular-tips.com/images/transclusionscopes/diag4.jpg

(The normal arrow is for new isolated scopes and the dashed is for new non-isolated scopes)

On the other hand, you can get the behavior you expected when you opened this article, that is the transcluded html using the isolated scope. I know you smart and you figured it out, but there it is:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | app.directive('person', function() {  return {  restrict: 'EA',  scope: {  header: '='  },  transclude:true,  link: function(scope, element, attrs, ctrl, transclude) {  scope.person = {  name: 'Directive Joe',  profession: 'Scope guy'  };  scope.header = 'Directive\'s header';  transclude(scope, function(clone, scope) {  element.append(clone);  });  }  };  }); |

**Try it**

Now it is using the isolated scope as the transcluded html scope.

Take in mind that maybe the people that will consume your directives are not aware that you’re tweaking the transcluded scope, so if you use it, be sure you document it well.

Posted by Jesus Rodriguez Mar 8th, 2014 [directives](http://angular-tips.com/blog/categories/directives/), [scopes](http://angular-tips.com/blog/categories/scopes/)