AngularJS + JQuery : How to get dynamic content working in angularjs

I am working on a Ajax app using both jQuery and AngularJS.

When I update content (which contains AngularJS bindings) of a div using jQuery's `html` function, the AngularJS bindings doesn't work.

Here is a JSFiddle version of what I am trying to do.

I have dynamic content inside a div with the ID `#dynamicContent` , and I have a refresh button that would update contents of this div when refresh is clicked. Increment works as expected if I don't refresh the content, but after I refresh, the AngularJS binding stops working.

This may not be valid in AngularJS, but I initially built application with jQuery and started using AngularJS later on so I can't migrate everything to AngularJS. Any help with getting this working in AngularJS is greatly appreciated.

jquery angularjs

edited Dec 9 '14 at 19:18
Chris Krycho
1,310 1 8 21

asked Aug 2 '12 at 5:25
Raja
374 3 6 14

1 Is there any particular reason for using JQuery for this functionality? As this is nicely and easily covered by angular: <jsfiddle.net/pkozlowski_opensource/YCrFD/2/>; – pkozlowski.opensource Aug 2 '12 at 5:48

3 This is just a simplified version of my real use case to show the problem. In actual application dynamic content is generated by grails taglib which is passed over to jquery as html. So I can't port all logic in grails taglib over to angularjs to make it pure angularjs. – Raja Aug 2 '12 at 14:57

@pkozlowski.opensource, that link is dead? Also you should join outdoors.stackexchange.com :) – Liam Feb 14 '14 at 11:09

3 Answers

You need to call `$compile` on the HTML string before inserting it into the DOM so that angular gets a chance to perform the binding.

In your fiddle, it would look something like this.

```
$("#dynamicContent").html(
  $compile(
    "<button ng-click='count = count + 1' ng-init='count=0'>Increment</button><span>count:
{{count}} </span>"
  )(scope)
);
```

Obviously, `$compile` must be injected into your controller for this to work.

Read more in the `$compile` documentation.

answered Aug 2 '12 at 15:17
Noah Freitas
4,931 3 23 49

2 Thanks Noah, compile is working when I do it in a controller method and directly bind that controller method to click event of button. Here is working example. But I in my actual app, I had to call the controller method from a different jquery function. so I grabbed reference to scope object and called refresh method to recompile which doesn't work. Here is example. Can you help with getting this example to work. – Raja Aug 2 '12 at 20:13

1 @Raja When you call angular methods/expression outside of the angular framework you should call $scope.$apply() to perform proper scope life-cycle of exception handling, executing watches, etc. jsfiddle.net/fABdD/14 – Artem Andreev Aug 3 '12 at 4:59

1 @ArtemAndreev exactly right. @Raja you could also move the call to `$scope.$apply()` into the `refresh()` function itself if it makes sense for your architecture: jsfiddle.net/nfreitas/8xkeh/1 – Noah Freitas Aug 3 '12 at 15:21

3 The link to `angular-1.0.1.min.js` in the original examples was 404. Here are updated working versions of @ArtemAndreev-s example: jsfiddle.net/pmorch/fABdD/186 and @NoahFreitas example: jsfiddle.net/pmorch/8xkeh/43 – Peter V. Mørch Dec 19 '13 at 0:21

1 Using $compile and manipulating DOM within a controller will lead you down a path of having untestable code. Directives should be used for changing the DOM never a controller. – Enzey Dec 3 '14 at 16:08

Another Solution in Case You Don't Have Control Over Dynamic Content

This works if you didn't load your element through a directive (ie. like in the example in the commented jsfiddles).

Wrap up Your Content

Wrap your content in a div so that you can select it if you are using **JQuery**. You an also opt to use native javascript to get your element.

```
<div class="selector">
    <grid-filter columnname="LastNameFirstName" gridname="HomeGrid"></grid-filter>
</div>
```

Use Angular Injector

You can use the following code to get a reference to $compile if you don't have one.

```
$(".selector").each(function () {
    var content = $(this);
    angular.element(document).injector().invoke(function($compile) {
        var scope = angular.element(content).scope();
        $compile(content)(scope);
    });
});
```

Summary

The original post seemed to assume you had a $compile reference handy. It is obviously easy when you have the reference, but I didn't so this was the answer for me.

One Caveat of the previous code

If you are using a asp.net/mvc bundle with minify scenario you will get in trouble when you deploy in release mode. The trouble comes in the form of Uncaught Error: [$injector:unpr] which is caused by the minifier messing with the angular javascript code.

Here is the way to remedy it:

Replace the prevous code snippet with the following overload.

```
...
angular.element(document).injector().invoke(
[
    "$compile", function($compile) {
        var scope = angular.element(content).scope();
        $compile(content)(scope);
    }
]);
...
```

This caused a lot of grief for me before I pieced it together.

edited Nov 7 '14 at 12:27

answered Jun 5 '14 at 11:15

jwize
1,583 13 24

Thanks as the above code worked for me scope and the explanation about compile being available. sometimes you miss the easy parts :) – Abs Aug 12 '14 at 11:52

I had to $digest after the compile. – Knu Sep 12 '14 at 13:14

1 Absolute lifesaver, I added a conditional check for angular in my code which allows it to use all the angular goodness when plugged into an angular app – LinuxN00b Dec 3 '14 at 10:43

Thanks,

This answer helped also for me. Only I had to add $apply on the scope, to manage to work. Is this the correct way ?

```
$(".selector").each(function () {
    var content = $(this);
    angular.element(document).injector().invoke(function($compile) {
        var scope = angular.element(content).scope();
```

```
        $compile(content)(scope);
        **scope.$apply();**
    });
});
```

answered Apr 26 at 13:21

Ralf D'hooge
31 3