

Can I make a function available in every controller in angular?

Asked 6 years, 5 months ago Active 2 years ago Viewed 138k times



190



61

If I have a utility function `foo` that I want to be able to call from anywhere inside of my `ng-app` declaration. Is there some way I can make it globally accessible in my module setup or do I need to add it to the scope in every controller?

angularjs

function

global

edited Jul 30 '15 at 16:18



Damjan Pavlica

11.6k 5 33 54

asked Feb 22 '13 at 14:05



Ludwig Magnusson

6,819 8 26 46

I am not 100% sure about this, but there is a chance you can also define it on your module like this: `module.value('myFunc', function(a){return a;});` and then inject it by name in your controllers. (If one wants to avoid making a service) – [user2173353](#) Oct 2 '14 at 15:51

Meaning that I have to add it to every controller manually. `$rootScope` is the way to go for what I wanted to do almost 2 years ago =) – [Ludwig Magnusson](#) Oct 3 '14 at 10:55

OK. :) I just use directives with isolated scope more often than plain controllers and I have to inject everything anyway. I like the modular code style that this provides. Also, you don't have to mess with parent scopes in any way and you don't have to search much for where your scope variables come from. :) – [user2173353](#) Oct 3 '14 at 12:13

5 Answers



290



You basically have two options, either define it as a service, or place it on your root scope. I would suggest that you make a service out of it to avoid polluting the root scope. You create a service and make it available in your controller like this:

```
<!doctype html>
<html ng-app="myApp">
<head>
  <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
  <script src="http://code.angularjs.org/1.1.2/angular.min.js"></script>
  <script type="text/javascript">
    var myApp = angular.module('myApp', []);
```

```

myApp.factory('myService', function() {
    return {
        foo: function() {
            alert("I'm foo!");
        }
    };
});

myApp.controller('MainCtrl', ['$scope', 'myService', function($scope, myService) {
    $scope.callFoo = function() {
        myService.foo();
    }
}]);
</script>
</head>
<body ng-controller="MainCtrl">
    <button ng-click="callFoo()">Call foo</button>
</body>
</html>

```

If that's not an option for you, you can add it to the root scope like this:

```

<!doctype html>
<html ng-app="myApp">
<head>
    <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
    <script src="http://code.angularjs.org/1.1.2/angular.min.js"></script>
    <script type="text/javascript">
        var myApp = angular.module('myApp', []);

        myApp.run(function($rootScope) {
            $rootScope.globalFoo = function() {
                alert("I'm global foo!");
            };
        });

        myApp.controller('MainCtrl', ['$scope', function($scope){

        }]);
    </script>
</head>
<body ng-controller="MainCtrl">
    <button ng-click="globalFoo()">Call global foo</button>
</body>
</html>

```

That way, all of your templates can call `globalFoo()` without having to pass it to the template from the controller.

edited Apr 28 '15 at 6:29

answered Feb 22 '13 at 14:29



Anders Ekdahl

19.8k 3 61 57

- 5 In the first solution, what if there are tons of `foo()` functions? Making a `$scope.callFoo()` wrapper for every one of them is too much work. How can I "attach" all functions of a library into scope so that it can be used in the template? I have a big unit conversion library that I want it to be available on my template. – AlexStack Oct 20 '14 at 18:41
- 3 My question as well. I tried this out and it works: you can "attach" it by saying `$scope.callFoo = myService.foo;` in place of creating a new wrapper in each place you want to use it. – Fitter Man Feb 12 '15 at 21:43
- 1 Thanks for your help, I was wondering how to make available a change language function throughout my app and `$rootScope` did the job. I wanted to keep the translation module apart from the application so I can plug it into other apps as well. – Paulo Pedroso May 23 '15 at 13:29

I'd suggest use an Angular Value rather than a Factory for this specific case unless you plan on having multiple functions within a service. If it's only one function, make it a Value. – Nicholas Blasgen Jan 3 '16 at 4:58

I got Error: [Injector:unpr] – Mark Thien Jun 25 '16 at 0:43

You can also combine them I guess:

53

```
<!doctype html>
<html ng-app="myApp">
<head>
  <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
  <script src="http://code.angularjs.org/1.1.2/angular.min.js"></script>
  <script type="text/javascript">
    var myApp = angular.module('myApp', []);

    myApp.factory('myService', function() {
      return {
        foo: function() {
          alert("I'm foo!");
        }
      };
    });

    myApp.run(function($rootScope, myService) {
      $rootScope.appData = myService;
    });
```

```
myApp.controller('MainCtrl', ['$scope', function($scope){
    }]);
```

```
</script>
</head>
<body ng-controller="MainCtrl">
  <button ng-click="appData.foo()">Call foo</button>
</body>
</html>
```

answered Jul 12 '14 at 15:55



ric

531 4 7

7 I think this should be the correct answer, justified by the answer of @Praym. It doesn't make sense specify a service dependency in 10 different controllers. – [jvannistelrooy](#) Dec 24 '14 at 10:21

Can the service include methods/functions which can CRUD properties/variables in the \$rootScope ? – [jezmck](#) Aug 5 '15 at 8:39



44



Though the first approach is advocated as 'the angular like' approach, I feel this adds overheads.

Consider if I want to use this myservice.foo function in 10 different controllers. I will have to specify this 'myService' dependency and then \$scope.callFoo scope property in all ten of them. This is simply a repetition and somehow violates the DRY principle.

Whereas, if I use the \$rootScope approach, I specify this global function gobalFoo only once and it will be available in all my future controllers, no matter how many.

answered Dec 19 '13 at 23:33

community wiki
[Praym](#)

5 There may be some value in the controllers 'documenting' where they get that global service call. If you were you yank one of your controllers into another application it would be less clear where that global function came from. I hear recognize your argument though. – [Matthew Payne](#) Apr 11 '14 at 18:03

It only has to be put on the scope if you need to call it from the view. In the controller, you can call it directly from the service. – [mcv](#) Nov 6 '15 at 12:13

10 This is a comment not an answer – [Elliott](#) Nov 27 '15 at 15:55

\$rootScope variable always null on page refresh in that case you will not get the function. That's why its good to inject the service and use its reference in application. – [Ehsan Hafeez](#) Apr 5 '17 at 14:54



4

AngularJs has "**Services**" and "**Factories**" just for problems like yours. These are used to have something global **between Controllers, Directives, Other Services or any other angularjs components**.. You can defined functions, store data, make calculate functions or whatever you want inside **Services** and use them in AngularJs Components as **Global**.like



```
angular.module('MyModule', [...])
  .service('MyService', ['$http', function($http){
    return {
      users: [...],
      getUserFriends: function(userId){
        return $http({
          method: 'GET',
          url: '/api/user/friends/' + userId
        });
      }
    }
  }])
```

if you need more

[Find More About Why We Need AngularJs Services and Factories](#)

edited Apr 27 '16 at 22:25

answered Apr 27 '16 at 21:50



[Hazarapet Tunanyan](#)

1,851 15 22



0

I'm a bit newer to Angular but what I found useful to do (and pretty simple) is I made a global script that I load onto my page before the local script with global variables that I need to access on all pages anyway. In that script, I created an object called "globalFunctions" and added the functions that I need to access globally as properties. e.g. `globalFunctions.foo = myFunc();` . Then, in each local script, I wrote `$scope.globalFunctions = globalFunctions;` and I instantly have access to any function I added to the globalFunctions object in the global script.

This is a bit of a workaround and I'm not sure it helps you but it definitely helped me as I had many functions and it was a pain adding all of them to each page.


edited Jul 25 '17 at 13:40

answered Jul 25 '17 at 1:54



izzy

71 4

1 I'm just curious why the downvotes? I am new and would like to know from the pros. – [izzy](#) Jul 30 '17 at 15:20 

Seems like a working solution to me. The only thing I would advocate, in order to make sure your scope is sufficiently isolated, is to create a global root Javascript utility class and hang your utility methods off of that so that you don't accidentally step on some other function name in the vast sea of things injected by Angular. – [J E Carter II](#) Mar 29 at 19:53

One thing to note, and maybe why this is downvoted, you could only use those functions in your templates, not your .ts modules as your function calls would not resolve at compile time. This is the reason for doing it the "angular" way. But, if you just want to add decorators and such to your templates, a global utility file is simple and fine. – [J E Carter II](#) Mar 29 at 20:03
