Event in AngularJS

http://www.bennadel.com/blog/2448-using-jquery-event-delegation-in-angularjs.htm

$scope.$on
$scope.$emit

http://stackoverflow.com/questions/14502006/working-with-scope-emit-and-on

Working with $scope.$emit and .$on

505
301

How can I send my `$scope` object from one controller to another using `.$emit` and `.$on`methods?

```
function firstCtrl($scope){
    $scope.$emit('someEvent',[1,2,3]);}function secondCtrl($scope){
    $scope.$on('someEvent',function(mass){ console.log(mass);});}
```

It doesn't work the way I think it should. How do `$emit` and `$on` work?

javascript angularjs

shareimprove this question	edited Dec 6 '14 at 13:16 rahilwazir **5,708**111933	asked Jan 24 '13 at 13:03 Paul Kononenko **2,665**4810

add a comment

9 Answers

activeoldestvotes

1078

First of all, parent-child scope relation does matter. You have two possibilities to emit some event:

- `$broadcast` -- dispatches the event downwards to all child scopes,
- `$emit` -- dispatches the event upwards through the scope hierarchy.

I don't know anything about your controllers (scopes) relation, but there are several options:

1. If scope of `firstCtrl` is parent of the `secondCtrl` scope, your code should work by replacing `$emit` by `$broadcast` in `firstCtrl`:

```
function firstCtrl($scope){
    $scope.$broadcast('someEvent',[1,2,3]);}function secondCtrl($scope){
    $scope.$on('someEvent',function(event, mass){ console.log(mass);});}
```

2. In case there is no parent-child relation between your scopes you can inject `$rootScope` into the controller and broadcast the event to all chil d scopes (i.e. also `secondCtrl`).

```
function firstCtrl($rootScope){
    $rootScope.$broadcast('someEvent',[1,2,3]);}
```

3. Finally, when you need to dispatch the event from child controller to scopes upwards you can use `$scope.$emit`. If scope of `firstCtrl` is pa rent of the `secondCtrl` scope:

```
function firstCtrl($scope){
    $scope.$on('someEvent',function(event, data){ console.log(data);});}function secondCtrl($scope){
    $scope.$emit('someEvent',[1,2,3]);}
```

shareimprove this answer	edited Dec 8 '14 at 9:55	answered Jan 24 '13 at 13:41 zbynour **13.4k**21627

6	Is there a way to fire an event from a service to a controller? – Zlatko Jan 16 '14 at 23:14
16	Yes theoretically you could inject `$rootScope` into your service and broadcast the event from the service. – zbynour Jan 17 '14 at 7:19
11	@Zlatko I'm pretty sure services by default are scope-less, and you need a scope to participate in the event system. So you somehow need to provide a scope to your service. $rootScope is the most general-purpose solution to that, but if you want your service to send events from a different scope, your controller could pass its scope to the service by setting a property on the service, and now the service can use the controller's scope. A more straight-forward technique might be for the controller to provide a function to the service which the service can call directly. – Oran Dennison Feb 21 '14 at 20:46
1	If you are using a iframe this article will be helpful charemza.name/blog/posts/angularjs/iframe/… – leticiaSep 21 '14 at 1:29
5	i made this plunkr based on @Zlatko's comment, and it works like a charm. It seems much less heavy-handed than passing in `$rootScope`. – sean9999 Nov 30 '14 at 6:17

show **9** more comments

87

I would additionally suggest a 4th option as a better alternative to the proposed options by @zbynour.

Use `$rootScope.$emit` rather than `$rootScope.$broadcast` regardless of the relationship between trasmitting and receiving controller. That way, the event remains within the set of `$rootScope.$$listeners` whereas with `$rootScope.$broadcast` the event propagates to all children scopes, most of which will probably not be listeners of that event anyway. And of course in the receiving controller's end you just use `$rootScope.$on`.

shareimprove this answer	answered Apr 6 '14 at 19:34 Thalis K. **2,462**11634

1	This would then basically serve as a central event bus correct? – jusopi Jun 11 '14 at 3:30
3	In a sense yes, the benefit being that you avoid event propagation. – Thalis K. Jun 11 '14 at 9:26
3	@ThalisK. thanks for this option. It avoids the propagation but on the other hand it requires `$rootScope`injection into controllers (what is not needed in general). But surely another option, thx! – zbynour Jul 2 '14 at 13:18
	you guys should choose this, this is simple and work all the time. All the time, you should prefer simple and easy to manage rather than smart code. – nXqd Oct 31 '14 at 11:39
44	Beware that $rootScope lives forever. If your controller is run twice, any $rootScope.$on inside it will be run twice, and caught events will result in a callback invoked twice. If you use $scope.$on instead, the callback will be destroyed along with your controller implicitly by AngularJS. – Filip Sobczak Nov 28 '14 at 16:09

add a comment

How can I send my $scope object from one controller to another using .$emit and .$on methods?

You can send any object you want within the hierarchy of your app, including **$scope**.

Here is a quick idea about how **broadcast** and **emit** work.

Notice the nodes below; all nested within node 3. You use **broadcast** and **emit** when you have this scenario.

Note: The number of each node in this example is arbitrary; it could easily be the number one; the number two; or even the number 1,348. Each number is just an identifier for this example. The point of this example is to show nesting of Angular controllers/directives.

```
3--------||-------1|2|------------||||||||
```

Check out this tree. How do you answer the following questions?

Note: There are other ways to answer these questions, but here we'll discuss **broadcast** and **emit**. Also, when reading below text assume each number has it's own file (directive, controller) e.x. one.js, two.js, three.js.

How does node **1** speak to node **3**?

In file **one.js**

```
scope.$emit('messageOne', someValue(s));
```

In file **three.js** - the uppermost node to all children nodes needed to communicate.

```
scope.$on('messageOne', someValue(s));
```

How does node 2 speak to node 3?

In file **two.js**

```
scope.$emit('messageTwo', someValue(s));
```

In file **three.js** - the uppermost node to all children nodes needed to communicate.

```
scope.$on('messageTwo', someValue(s));
```

How does node 3 speak to node 1 and/or node 2?

In file **three.js** - the uppermost node to all children nodes needed to communicate.

```
scope.$broadcast('messageThree', someValue(s));
```

In file **one.js** && **two.js** whichever file you want to catch the message or both.

```
scope.$on('messageThree', someValue(s));
```

How does node 2 speak to node 1?

In file **two.js**

```
scope.$emit('messageTwo', someValue(s));
```

28

In file **three.js** - the uppermost node to all children nodes needed to communicate.

```
scope.$on('messageTwo',function(event, data ){
  scope.$broadcast('messageTwo', data );});
```

In file **one.js**

```
scope.$on('messageTwo', someValue(s));
```

HOWEVER

When you have all these nested child nodes trying to communicate like this, you will quickly see many **$on's**, **$broadcast's**, and **$emit's**.

Here is what I like to do.

In the uppermost PARENT NODE (**3** in this case...), which may be your parent controller...

So, in file **three.js**

```
scope.$on('pushChangesToAllNodes',function(event, message ){
  scope.$broadcast( message.name, message.data );});
```

Now in any of the child nodes you only need to **$emit** the message or catch it using **$on**.

NOTE: It is normally quite easy to cross talk in one nested path without using **$emit**, **$broadcast**, or **$on**, which means most use cases are for when you are trying to get node **1** to communicate with node **2** or vice versa.

How does node 2 speak to node 1?

In file **two.js**

```
scope.$emit('pushChangesToAllNodes', sendNewChanges());function sendNewChanges(){// for some event.return{ name:'talkToOne', data:[1,2,3]};}
```

In file **three.js** - the uppermost node to all children nodes needed to communicate.

We already handled this one remember?

In file **one.js**

```
scope.$on('talkToOne',function(event, arrayOfNumbers ){
  arrayOfNumbers.forEach(function(number){
    console.log(number);});});
```

You will still need to use **$on** with each specific value you want to catch, but now you can create whatever you like in any of the nodes without having to worry about how to get the message across the parent node gap as we catch and broadcast the generic **pushChangesToAllNodes**.

Hope this helps...

shareimprove this answer	edited Oct 7 at 16:25	answered Jul 11 at 14:47 SoEzPz **1,572**1225

	how to decide which one is 3,2 and 1? – Jayesh Jain Sep 18 at 13:58
	The 3, 2, and 1 are either nested controllers or directives. As you create your app, keep in mind your nesting and apply the logic above. For an example, we could say 3 is the $rootScope of the application; and everything is nested below it. 3, 2, and 1 are arbitrary. – SoEzPz Sep 18 at 16:00
	Great examples! But I'm still thinking that better to use **own** event-dispatcher in parent to communicate group of controllers. Also useful to keep dispatcher's creat ion as service to use it as pattern. – DenisKolodin Oct 20 at 10:31