how to stop Javascript forEach? [duplicate]

Asked 8 years, 4 months ago Active 1 year, 2 months ago Viewed 387k times

328

95

This question already has an answer here:
Short circuit Array.forEach like calling break *28 answers*

i'm playing with nodejs and mongoose — trying to find specific comment in deep comments nesting with recursive func and foreach within. Is there a way to stop nodejs forEach? As i understand every forEach iteration is a function and and i can't just do "break",only "return" but this won't stop foreach.

```
function recurs(comment){
    comment.comments.forEach(function(elem){
        recurs(elem);
        //if(...) break;
    });
}
```

javascript ecmascript-5

edited Aug 13 '12 at 14:56
Charles
46.6k 12 90 128

asked Jun 7 '11 at 5:00
kulebyashik
1,783 3 13 12

marked as duplicate by Bergi javascript Jul 25 '17 at 15:47

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

13 Answers

831

You can't break from a `forEach`. I can think of three ways to fake it, though.

1. The Ugly Way: pass a second argument to `forEach` to use as context, and store a boolean in there, then use an `if`. This looks awful.

2. The Controversial Way: surround the whole thing in a `try-catch` block and throw an exception when you want to break. This looks pretty bad and may affect performance, but can be encapsulated.

3. The Fun Way: use `every()`.

```
['a', 'b', 'c'].every(function(element, index) {
  // Do your thing, then:
  if (you_want_to_break) return false
  else return true
})
```

You can use `some()` instead, if you'd rather `return true` to break.

edited Jul 5 '17 at 19:44
Vadim Kotov
5,587 7 36 49

answered Jun 7 '11 at 5:18
slezica
48k 18 84 140

47 +1, though it sounds more natural to me to use `some()` and `return true` when you want to break. – Giacomo Jun 7 '11 at 7:54

19 Or more elegantly, put `return !you_want_to_break` inside the loop instead of the if..else block. Saves two lines. :-) – sffc Nov 27 '13 at 23:25

14 `every` supported everywhere except IE7 & 8 (I had to look it up, so thought I'd share) – jbobbins Sep 19 '14 at 0:55

5 There should be number 4. The Beautiful Way: use `some()` – Preexo Oct 21 '14 at 3:59

5 Unfortunately using every() or some() will not solve cases where you would like to break on a specific value and return it. ECMAScript2015 new for...of (developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...) could help with that part but the drawback is that this solution can cause even more issues with older browsers. If you are willing to change route completely and use a different more generic approach, a solution like this (github.com/nbouvrette/forEach) could help you and potentially solve even other headaches. – Nicolas Bouvrette Jun 13 '16 at 0:14

Breaking out of Array#forEach is not possible. (You can inspect the source code that implements it in Firefox on the linked page, to confirm this.)

```
function recurs(comment) {
    for (var i = 0; i < comment.comments.length; ++i) {
        var subComment = comment.comments[i];
        recurs(subComment);
        if (...) {
            break;
        }
    }
}
```

(or, if you want to be a little more clever about it and `comment.comments[i]` is always an object:)

```
function recurs(comment) {
    for (var i = 0, subComment; subComment = comment.comments[i]; ++i) {
        recurs(subComment);
        if (...) {
            break;
        }
    }
}
```

answered Jun 7 '11 at 5:12

Domenic
82.9k 31 190 244

It is possible by throwing inside forEach function, as the accepted answer states. – Estus Flask Sep 23 '16 at 13:55

1 sure is, but using exceptions to control program flow is a bad bad practice – Igor Donin Feb 16 '18 at 18:15

33

In some cases `Array.some` will probably fulfil the requirements.

edited May 14 '15 at 17:15

rink.attendant.6
19.5k 18 70 126

answered Oct 4 '12 at 10:03

igor
1,643 19 18

This should be the canonical answer as it will actually stop processing once it finds the correct element. While `forEach` and `every` (as I understand it)

how is the browser support ? – imal hasaranga perera Apr 1 '17 at 14:10

@imalhasarangaperera Here you can find current browser support. developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/… – Senthe Apr 21 '17 at 12:15

29

As others have pointed out, you can't cancel a `forEach` loop, but here's my solution:

```
ary.forEach(function loop(){
    if(loop.stop){ return; }

    if(condition){ loop.stop = true; }
});
```

Of course this doesn't actually break the loop, it just prevents code execution on all the elements following the "break"

answered Jun 7 '11 at 5:26

Mark Kahn

66.1k 23 145 197

1 I like this one. I would just combine the last line to `loop.stop = condition` though. It shouldn't make any difference because when it's set to `true` it won't be run anymore. – pimvdb Jun 7 '11 at 8:55

2 Clever use of named function expressions – Raynos Jun 7 '11 at 9:26

9 I don't think this solution is a good idea. imagine you are looping 10000 elements and your condition is stop at second element, then you are going to do unnecessary iteration of 9998 times for nothing. The best approaches are either using `some` or `every` . – Ali Mar 7 '14 at 18:19

1 @Ali zyklus said that.... it's another, valid, solution! Depends on your case.... Why do people spend time on critisizing instead of presenting fresh new different solutions...?!?!? – Pedro Ferreira Mar 28 '16 at 10:36

3 @PedroFerreira my argument is valid. I'm not offending anybody. We need to be criticized about our code to make it better. I would rather read other implementation and help them improve it than reinvent the wheel. – Ali Mar 29 '16 at 19:35

I guess you want to use Array.prototype.find Find will break itself when it finds your specific value in the array.

```
  {name: 'cherries', quantity: 5}
];

function findCherries(fruit) {
  return fruit.name === 'cherries';
}

console.log(inventory.find(findCherries));
// { name: 'cherries', quantity: 5 }
```

answered Jul 11 '17 at 7:02

Shrihari Balasubramani
485 1 5 12

8

forEach does not break on return, there are ugly solutions to get this work but I suggest not to use it, instead try to use `Array.prototype.some` or `Array.prototype.every`

```
var ar = [1,2,3,4,5];

ar.some(function(item,index){
  if(item == 3){
     return true;
  }
  console.log("item is :"+item+" index is : "+index);
});
```

Run code snippet Expand snippet

edited Aug 8 '18 at 4:19

answered Apr 20 '17 at 16:35

imal hasaranga perera
5,378 2 34 29

it works but it will iterate all the member of the array – Ankur Shah Jul 24 '18 at 6:31

sorry there was a mistake, you need to do return true – imal hasaranga perera Aug 8 '18 at 4:20

5

You can use Lodash's `forEach` function if you don't mind using 3rd party libraries.

Example:

```
var _ = require('lodash');

_.forEach(comments, function (comment) {
    do_something_with(comment);

    if (...) {
        return false;     // Exits the loop.
    }
})
```

answered Oct 21 '16 at 12:17

exmaxx
1,268 14 15

3

```
    var f = "how to stop Javascript forEach?".split(' ');
    f.forEach(function (a,b){
        console.info(b+1);
        if (a == 'stop') {
            console.warn("\tposition: \'stop\'["+(b+1)+"] \r\n\tall length: " +
(f.length));
            f.length = 0; //<--!!!
        }
    });
```

edited Jan 10 '16 at 12:24

answered Jan 9 '16 at 17:10

Владимир
31 2

Nice one! I'd use third argument on callback though and length on it. – igor Apr 27 '17 at 13:23

3

1) use a old `for` loop: this will be the most compatible solution but can be very hard to read when used often in large blocks of code:

```
var testArray = ['a', 'b', 'c'];
for (var key = 0; key < testArray.length; key++) {
    var value = testArray[key];
    console.log(key); // This is the key;
    console.log(value); // This is the value;
}
```

2) use the new ECMA6 (2015 specification) in cases where compatibility is not a problem. Note that even in 2016, only a few browsers and IDEs offer good support for this new specification. While this works for iterable objects (e.g. Arrays), if you want to use this on non-iterable objects, you will need to use the `Object.entries` method. This method is scarcely available as of June 18th 2016 and even Chrome requires a special flag to enable it: `chrome://flags/#enable-javascript-harmony` . For Arrays, you won't need all this but compatibility remains a problem:

```
var testArray = ['a', 'b', 'c'];
for (let [key, value] of testArray.entries()) {
    console.log(key); // This is the key;
    console.log(value); // This is the value;
}
```

3) A lot of people would agree that neither the first or second option are good candidates. Until option 2 becomes the new standard, most popular libraries such as AngularJS and jQuery offer their own loop methods which can be superior to anything available in JavaScript. Also for those who are not already using these big libraries and that are looking for lightweight options, solutions like this can be used and will almost be on par with ECMA6 while keeping compatibility with older browsers.

edited Jun 18 '16 at 13:02

answered Jun 18 '16 at 2:34

Nicolas Bouvrette

1,470 1 17 35

1

Below code will break the foreach loop once the condition is met, below is the sample example

```
var array = [1,2,3,4,5];
var newArray = array.slice(0,array.length);
array.forEach(function(item,index){
    //your breaking condition goes here example checking for value 2
```

```
    })
    array = newArray;
```

answered Jun 13 '17 at 4:01

D G ANNOJIRAO

52 4

does it will not create a memory leakage problem? – Jasti Sri Radhe Shyam Jul 23 '18 at 21:00

definitely it will not create memory leakage, because we are setting the original array with same exact data as it is, and it will not be dangling pointer, if you have still concern about newArray variable which was used above, after reassigning array = newArray you can set newArray= null – D G ANNOJIRAO Jul 24 '18 at 15:59

-3

You can break from a forEach loop if you overwrite the Array method:

```
(function(){
    window.broken = false;

        Array.prototype.forEach = function(cb, thisArg) {
            var newCb = new Function("with({_break: function(){window.broken = true;}})
{("+cb.replace(/break/g, "_break()")+"(arguments[0], arguments[1], arguments[2]));}");
            this.some(function(item, index, array){
                 newCb(item, index, array);
                 return window.broken;
            }, thisArg);
            window.broken = false;
        }

}())
```

example:

```
[1,2,3].forEach("function(x){\
    if (x == 2) break;\
    console.log(x)\
}")
```

Unfortunately with this solution you can't use normal break inside your callbacks, you must wrap invalid code in strings and native functions don't work directly (but you can work around that)

Happy breaking!

answered May 16 '16 at 22:23

ApplePear
11 1

3 Never extend prototype, you might break things without knowing ant it will break your mind! Such a bad solution – Lukas Sep 20 '16 at 10:54

-10

Wy not use plain return?

```
function recurs(comment){
comment.comments.forEach(function(elem){
    recurs(elem);
    if(...) return;
});
```

it will return from 'recurs' function. I use it like this. Althougth this will not break from forEach but from whole function, in this simple example it might work

answered Mar 12 '14 at 18:17

morterad
1

You can not return in a loop – Anonymoose Mar 12 '14 at 18:36

1 @Hazaart Although you can't do this in `forEach`, you can return in loops and the function will exit with no further iteration of the loop. (e.g. `for (... in ...)`, `while`, etc.) – Sung Cho Jul 9 '15 at 2:09

jQuery provides an `each()` method, not `forEach()`. You can break out of `each` by returning `false`. `forEach()` is part of the ECMA-262 standard, and the only way to break out of that that I'm aware of is by throwing an exception.

```
function recurs(comment) {
  try {
    comment.comments.forEach(function(elem) {
      recurs(elem);
      if (...) throw "done";
    });
  } catch (e) { if (e != "done") throw e; }
}
```

Ugly, but does the job.

answered Jun 7 '11 at 5:22

Joe Taylor
534 4 12

25 -1 for spouting "use jquery" when the OP specifically said "node.js" – Mark Kahn Jun 7 '11 at 5:30

1 @cwolves I thought I saw "jQuery" in there somewhere. Guess not, but heh anyhow because jQuery and node.js can be used in conjunction with one another. – Joe Taylor Jun 7 '11 at 14:22

6 no, you can't use jQuery in node. If nothing else, there is an explicit reference to `window` which will throw an error if loaded in node. Then you have all of the wasted code: The entire Sizzle engine, the entire jQuery root function (literally -- there's a reference to `document` in it -- it ***expects*** a DOM), etc. The ONLY thing you can use from jQuery in node are a few of the helper functions, and those have clones in other libraries like underscore that are much better suited to node. – Mark Kahn Jun 7 '11 at 15:04

2 @cwolves Guess I'm getting rusty. I was thinking of Rhino. – Joe Taylor Jun 8 '11 at 2:15