How to remove item from array by value?

Asked 8 years, 11 months ago Active 1 month ago Viewed 896k times

817

164

Is there a method to remove an item from a JavaScript array?

Given an array:

```
var ary = ['three', 'seven', 'eleven'];
```

I would like to do something like:

```
removeItem('seven', ary);
```

I've looked into `splice()` but that only removes by the position number, whereas I need something to remove an item by its value.

javascript arrays

edited Sep 29 '17 at 12:45
Paul Roub
33.4k 8 62 77

asked Oct 17 '10 at 17:43
MacMac
13.6k 49 133 210

3 See also: Remove an array element by value in JavaScript and Remove specific element from an array? – Bergi Aug 11 '14 at 16:47

52 PLEASE USE --> `Array.filter()` – Eric Hodonsky Mar 20 '17 at 18:08

I wrote various solutions for this (remove one or multiple values) and this is my ultimate solution (I benchmarked and it is faster than lodash). Have a go: gist.github.com/ardeshireshghi/0d97db4ae09bc2f90609c536fc63c648 – Ardi Oct 4 '17 at 15:03

It can also be found here: stackoverflow.com/questions/5767325/… This is the benchmark: jsperf.com/array-without-benchmark-against-lodash – Ardi Oct 4 '17 at 16:36

35 Answers

1 2 next

443

This can be a global function or a method of a custom object, if you aren't allowed to add to native prototypes. It removes all of the items from the array that match any of the arguments.

```
Array.prototype.remove = function() {
    var what, a = arguments, L = a.length, ax;
    while (L && this.length) {
        what = a[--L];
        while ((ax = this.indexOf(what)) !== -1) {
            this.splice(ax, 1);
        }
    }
    return this;
};

var ary = ['three', 'seven', 'eleven'];

ary.remove('seven');

/*  returned value: (Array)
three,eleven
*/
```

To make it a global-

```
function removeA(arr) {
    var what, a = arguments, L = a.length, ax;
    while (L > 1 && arr.length) {
        what = a[--L];
        while ((ax= arr.indexOf(what)) !== -1) {
            arr.splice(ax, 1);
        }
    }
    return arr;
}
var ary = ['three', 'seven', 'eleven'];
removeA(ary, 'seven');


/*  returned value: (Array)
```

And to take care of IE8 and below-

```
if(!Array.prototype.indexOf) {
    Array.prototype.indexOf = function(what, i) {
        i = i || 0;
        var L = this.length;
        while (i < L) {
            if(this[i] === what) return i;
            ++i;
        }
        return -1;
    };
}
```

edited Nov 21 '12 at 5:38
Community ♦
1 1

answered Oct 17 '10 at 20:16
kennebec
83.5k 26 90 122

3 @xorinzor No, the .prototype property is cross-browser. – Casey Rodarmor Oct 5 '12 at 5:08

133 Never change Array prototype. Funny things starts to happen. – szanata May 7 '13 at 12:46

22 @madeinstefano, one or two examples of the funny (bad) things that would happen? – Majid Fouladpour Jul 23 '13 at 22:38

15 why do people show examples adding to the array prototype? stack overflow is for learning good practices – Blair Anderson Dec 6 '14 at 0:49

6 @YonnTrimoreau and what? define a non-enumerable property `Object.defineProperty(Array.prototype, "remove", {enumerable : false});` – naXa Aug 28 '15 at 9:38

1395

You can use the `indexOf` method like this:

```
var index = array.indexOf(item);
if (index !== -1) array.splice(index, 1);
```

> **Note**: You'll need to shim it for IE8 and below

edited Feb 8 '18 at 18:54

KyleMit
61.8k 40 273 436

answered Oct 17 '10 at 17:45

SLaks
720k 147 1696 1815

67 And loop on it while the index isn't `-1` – Colin Hebert Oct 17 '10 at 17:50

8 It would be best to do a check to only splice if different than `-1` , there are like millions of options, choose wisely jsperf.com/not-vs-gt-vs-ge/4 – ajax333221 May 29 '12 at 21:19

31 If you use jquery, you can use `$.inArray` instead of `indexOf` , which is cross browser compatible. – Tamás Pap May 21 '13 at 8:55

3 Check stackoverflow.com/questions/5767325/… – Dimuthu Oct 18 '13 at 5:50

32 Make sure index!==(-1) , i.e. item exists in array, or else you will splice out the last element in array. – Ronen Rabinovici Oct 27 '13 at 21:08

354

A one-liner will do it,

```
var ary = ['three', 'seven', 'eleven'];

// Remove item 'seven' from array
var filteredAry = ary.filter(function(e) { return e !== 'seven' })
//=> ["three", "eleven"]

// In ECMA6 (arrow function syntax):
var filteredAry = ary.filter(e => e !== 'seven')
```

This makes use of the filter function in JS. It's supported in IE9 and up.

What it does (from the doc link)

> filter() calls a provided callback function once for each element in an array, and constructs a new array of all the values for which callback returns a value that coerces to true. callback is invoked only for indexes of the array which have assigned values; it is not invoked for indexes which have been deleted or which have never been assigned values. Array elements which do not pass the callback test are simply skipped, and are not included in the new array.

So basically, this is the same as all the other `for (var key in ary) { ... }` solutions, except that the `for in` construct is supported as of

Basically, filter is a convenience method that looks a lot nicer (and is chainable) as opposed to the `for in` construct (AFAIK).

edited Aug 26 '16 at 12:30

SidOfc
3,930 3 21 42

answered Dec 29 '13 at 16:05

John Williams
6,118 3 28 42

5 Wondering this wonderful one-liner does not get more love. +1 No loops. One can add as many as values he want to remove by using `&&` for values. – user1299518 Aug 22 '14 at 16:15

@bamboon The filter function checks whether an element meets the criteria. The function is executed with each element one by one and upon returning `true`, the resulting array will retain the element. It is omitted on `false`. – Qwerty Oct 14 '14 at 14:29

1 This is really straightforward, with the caveat mentioned by @SLaks that it creates a copy of the array, and doesn't affect references to the variable. – tobylaroni May 19 '15 at 12:57

8 Note that this should be used as `array = array.filter()`, not just `array.filter()`. – Jeffrey Roosendaal Nov 30 '15 at 14:19

6 Updated answer with @JeffreyRoosendaal contribution (`filter() does not mutate the array on which it is called.` from developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/…). – Manolo Jan 14 '16 at 15:02

131

You can use underscore.js. It really makes things simple.

For example, with this:

```
var result = _.without(['three','seven','eleven'], 'seven');
```

And `result` will be `['three','eleven']`.

In your case the code that you will have to write is:

```
ary = _.without(ary, 'seven')
```

It reduces the code that you write.

edited May 31 '15 at 5:15

Julius Musseau

answered Feb 19 '13 at 9:52

yatsal

15 I never said don't use the library in other places. If the code looks cleaner then i don't mind including a library. Why do people use jquery , why not use raw javascript then? – vatsal Mar 4 '13 at 5:17

10 @vatsal - because library developers can concentrate on making the functions behind their library functions fast, concise and cross browser, while I get to concentrate on my application and its purpose. It saves me thinking time so that I have extra time to make the application better and not worrying about the small functions that makeup my application. Someone already invented the wheel, why would someone remake it every time they build a car? – styks May 13 '14 at 13:05

11 Hi Kelvin i totally agree with you. A person had a put a comment and he removed it later, he was saying that using libraries like underscore is not cool and we should not accept such answers. I was trying to answer it. – vatsal May 13 '14 at 13:37

Do note this will not modify the array in place; A new array will be returned instead. – gcscaglia Feb 22 '17 at 20:42

50

Check out this way:

```
for(var i in array){
    if(array[i]=='seven'){
        array.splice(i,1);
        break;
    }
}
```

and in a function:

```
function removeItem(array, item){
    for(var i in array){
        if(array[i]==item){
            array.splice(i,1);
            break;
        }
    }
}

removeItem(array, 'seven');
```

edited Jan 23 '17 at 14:24

answered Jul 17 '11 at 17:31

gadlol
893 2 11 22

splice, like so: i--. That's because you just shrunk the array and you'll end up skipping an element otherwise. – Doug S Oct 27 '12 at 5:44

3 To add to my above comment, the code would then be: for (var i = 0; i < array.length; i++) {/*etc...*/ array.splice(i,1); i--; – Doug S Oct 27 '12 at 5:49

35

Here's a version that uses jQuery's inArray function:

```
var index = $.inArray(item, array);
if (index != -1) {
    array.splice(index, 1);
}
```

edited Mar 23 '17 at 19:46

answered Jul 12 '13 at 9:02

CorayThan

9,135 19 81 134

17 Splice supports negative indices to take from the end, so if the item is not found, this code removes the last item from the array. – dmeglio Apr 21 '15 at 13:04

2 @dman2306 Huh, didn't see your comment until now, but that's a really good point. Fixed it for that issue. – CorayThan Mar 23 '17 at 19:46

34

You can do it with these two ways:

```
var arr = ["1","2","3","4"] // we wanna delete number "3"
```

first:

```
arr.indexOf('3') !== -1 && arr.splice(arr.indexOf('3'), 1)
```

second (ES6):

```
arr = arr.filter(e => e !== '3')
```

 1

3 What if element doesn't exist in the array, e.g. `a.indexOf('3') === -1` ? Will it change the array in some weird way (incorrect behavior) or won't do anything at all (correct behavior)? – izogfif Mar 14 '18 at 13:51

2 If the item doesn't exist in the first method, it removes the last item in the list – Luke Wenke Jul 24 '18 at 9:33

Absolutely, My choice is the second way – AmerllicA Jul 24 '18 at 13:07

@izogfif, I fix the code for your exception, Thanks a lot. – AmerllicA Jul 24 '18 at 13:11

28

```
var index = array.indexOf('item');

if(index!=-1){

   array.splice(index, 1);
}
```

answered Nov 13 '12 at 13:49

 Kld

5,319 3 28 45

16

What you're after is filter

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

This will allow you to do the following:

```
var ary = ['three', 'seven', 'eleven'];
var aryWithoutSeven = ary.filter(function(value) { return value != 'seven' });
console.log(aryWithoutSeven); // returns ['three', 'eleven']
```

This was also noted in this thread somewhere else: https://stackoverflow.com/a/20827100/293492

4 That will not mutate the array. – SLaks Aug 24 '14 at 2:16

8 No; that assigns the variable to point to a new object. If you have other references to the original array, they will not be affected. – SLaks Aug 24 '14 at 2:18

15

Simply

```
var ary = ['three', 'seven', 'eleven'];
var index = ary.indexOf('seven'); // get index if value found otherwise -1

 if (index > -1) { //if found
   ary.splice(index, 1);
 }
```

edited Nov 15 '18 at 9:31

answered Oct 29 '18 at 16:15

Matee Gojra

1,212 13 19

2 Thank you for this code snippet, which might provide some limited, immediate help. A proper explanation would greatly improve its long-term value by showing *why* this is a good solution to the problem, and would make it more useful to future readers with other, similar questions. Please edit your answer to add some explanation, including the assumptions you've made. – FrankerZ Oct 29 '18 at 18:02

@FrankerZ added comments parallel to code lines – Matee Gojra Nov 15 '18 at 9:33

14

The simplest solution is:

array - array for remove some element valueForRemove; valueForRemove - element for remove;

```
array.filter(arrayItem => !array.includes(valueForRemove));
```

More simple:

No pretty, but works:

```
array.filter(arrayItem => array.indexOf(arrayItem) != array.indexOf(valueForRemove))
```

No pretty, but works:

```
while(array.indexOf(valueForRemove) !== -1) {
  array.splice(array.indexOf(valueForRemove), 1)
}
```

edited Aug 23 at 12:06

answered Feb 26 at 20:45

Jackkobec

1,639 9 14

Thats not even close to the correct answer? You should edit this for the working example. It's confusing now. – Arnas Pecelis Mar 4 at 15:02

1 Thanks, code has been fixed. It was alias mistake. – Jackkobec Mar 5 at 15:49

while i like this approach because it uses immutable data structures, this isn't a catch-all solution because sometimes the developer *does* want to mutate the array in place – feihcsim Aug 15 at 15:43

The filter() method creates a new array with all elements that pass the test implemented by the provided function. See developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/… – Jackkobec Aug 15 at 16:11

12

Seeing as there isn't a pretty one, here's a simple and reusable ES6 function.

```
const removeArrayItem = (arr, itemToRemove) => {
  return arr.filter(item => item !== itemToRemove)
}
```

Usage:

```
const items = ['orange', 'purple', 'orange', 'brown', 'red', 'orange']
removeArrayItem(items, 'orange')
```

1,181 5 15

3 removeArrayItem method doesn't remove item from the array. It creates new array without item. – WebBrother Nov 27 '17 at 16:06

Correct @WebBrother. From a 'consumer' point of view, the implementation is not really my concern - if I give it an array and an item, I get an array back with the item removed, so the name makes sense to me. Thoughts? – Shakespeare Dec 8 '17 at 11:09

If you're going to use ES6, make sure you're using the correct comparison (avoid !== and instead use !Object.is(item, itemToRemove). – Sterling Bourne Jun 5 '18 at 20:13

11

ES6 way.

```
const commentsWithoutDeletedArray = commentsArray.filter(comment => comment.Id !==
commentId);
```

edited Nov 22 '18 at 15:17

answered May 29 '17 at 14:34

Oliver Dixon

4,064 5 39 68

I think mapreduce functions in javascript are pretty fast, but splice would still be faster. so a better implementation might be `const removeArrayItem = (arr, itemToRemove) => { return arr.includes(itemToRemove)? arr.splice(arr.indexOf(itemToRemove), 1): arr }` – roberto tomás Jul 26 '17 at 14:51

4 @robertotomás I prefer readability over slight performance gains. – Oliver Dixon Dec 1 '18 at 13:38

10

Really, i can't see why this can't be solved with

```
arr = arr.filter(value => value !== 'seven');
```

Or maybe you want to use vanilla JS

```
arr = arr.filter(function(value) { return value !== 'seven' });
```

rbenvenuto
1,227 2 8 16

Maybe because it doesn't remove the item from the array? – Z. Khullah Sep 3 '18 at 3:17

Yes, it doesn't, it creates a new array without the string `'seven'` – rbenvenuto Sep 3 '18 at 20:17

Perfect solution, thank you! – AlexioVay Sep 16 '18 at 17:01

9

If you have unique values in your array and ordering doesn't matter, you can use *Set*, and it has *delete*:

```
var mySet = new Set(['foo']);
mySet.delete('foo'); // Returns true.  Successfully removed.
mySet.has('foo');    // Returns false. The "foo" element is no longer present.
```

edited May 3 '18 at 21:04

Dan H
8,928 3 27 30

answered May 18 '17 at 14:05

greene
299 2 3

7

Removing all matching elements from the array (rather than just the first as seems to be the most common answer here):

```
while ($.inArray(item, array) > -1) {
    array.splice( $.inArray(item, array), 1 );
}
```

I used jQuery for the heavy lifting, but you get the idea if you want to go native.

answered Jan 22 '14 at 17:04

Jason
1,499 2 18 20

4 So you run $.inArray twice if an item is found, why not store the index on the first run and re-use that? #effectiveprogramming #resourcesdonotgrowontrees while((index = $.inArray(item, array)) > -1) array.splice(index, 1); – patrick May 29 '17 at 13:56

When you need to remove a value present multiple times in the array(e.g. [1,2,2,2, 4, 5,6]).

7

```
function removeFrmArr(array, element) {
  return array.filter(e => e !== element);
};
var exampleArray = [1,2,3,4,5];
removeFrmArr(exampleArray, 3);
// return value like this
//[1, 2, 4, 5]
```

You can use splice to remove a single element from the array but splice can't remove multiple similar elements from the array.

```
function singleArrayRemove(array, value){
  var index = array.indexOf(value);
  if (index > -1) array.splice(index, 1);
  return array;
}
var exampleArray = [1,2,3,4,5,5];
singleArrayRemove(exampleArray, 5);
// return value like this
//[1, 2, 3, 4, 5]
```

edited Apr 11 at 7:20

answered Feb 13 '18 at 7:27

Subhojit Mondal
237 4 13

1 You should consider adding some explanation to your code. – Zenoo Feb 13 '18 at 8:16

1 Line 5 should be `removeFrmArr(exampleArray, 3);` , current code throws a ReferenceError. – mikiqex Dec 25 '18 at 19:21

6

a very clean solution working in all browsers and without any framework is to asign a new Array and simply return it without the item you want to delete:

```
/**
 * @param {Array} array the original array with all items
 * @param {any} item the time you want to remove
```

```
var removeItemFromArray = function(array, item){
  /* assign a empty array */
  var tmp = [];
  /* loop over all array items */
  for(var index in array){
    if(array[index] !== item){
      /* push to temporary array if not like item */
      tmp.push(array[index]);
    }
  }
  /* return the temporary array */
  return tmp;
}
```

edited May 19 '17 at 5:44

dcordz
83 1 6

answered Jan 7 '16 at 13:52

mtizziani
698 8 18

+1 for a clean solution. Too many answers tend to suggest one 3rd party resource or another when sometimes we need a purer solution (and they are all great just not in all use cases). – Tahir Khalid Jun 29 '16 at 15:54

6

In all values unique, you can:

```
a = new Set([1,2,3,4,5]) // a = Set(5) {1, 2, 3, 4, 5}
a.delete(3) // a = Set(5) {1, 2, 4, 5}
[...a] // [1, 2, 4, 5]
```

edited Nov 23 '18 at 11:05

answered Nov 23 '18 at 10:58

Eugene Lyzo
71 1 2

4

`indexOf` is an option, but it's implementation is basically searching the entire array for the value, so execution time grows with array size. (so it is in every browser I guess, I only checked Firefox).

Basically you can use an object to make an index for your array, like so:

```
var index={'three':0, 'seven':1, 'eleven':2};
```

Any sane JavaScript environment will create a searchable index for such objects so that you can quickly translate a key into a value, no matter how many properties the object has.

This is just the basic method, depending on your need you may combine several objects and/or arrays to make the same data quickly searchable for different properties. If you specify your exact needs I can suggest a more specific data structure.

answered Oct 17 '10 at 18:32

aaaaaaaaaaaa
3,234 19 22

Mind that this is NOT clugy as an associative array is actually an object: var arr = []; arr['zero'] = 1, arr['one'] = 2; is equivalent to: {zero: 1, one: 2} – Cody Sep 11 '12 at 20:11

4

You can achieve this using **Lodash** `_.remove` function.

```
var array = ['three', 'seven', 'eleven'];
var evens = _.remove(array, function(e) {
  return e !== 'seven';
});

console.log(evens);


<script type="text/javascript"
src="https://cdn.jsdelivr.net/npm/lodash@4.17.10/lodash.min.js"></script>
```

Run code snippet Expand snippet

answered Sep 3 '18 at 9:11

The trick is to go through the array from end to beginning, so you don't mess up the indices while removing elements.

3

```
var deleteMe = function( arr, me ){
   var i = arr.length;
   while( i-- ) if(arr[i] === me ) arr.splice(i,1);
}

var arr = ["orange","red","black", "orange", "white" , "orange" ];

deleteMe( arr , "orange");
```

arr is now ["red", "black", "white"]

answered Mar 17 '15 at 17:43

chaoticflow
41 1

2

Non-destructive removal:

```
function removeArrayValue(array, value)
{
    var thisArray = array.slice(0); // copy the array so method is non-destructive

    var idx = thisArray.indexOf(value); // initialise idx

    while(idx != -1)
    {
        thisArray.splice(idx, 1); // chop out element at idx

        idx = thisArray.indexOf(value); // look for next ocurrence of 'value'
    }

    return thisArray;
}
```

answered Jun 25 '16 at 12:17

You can use `without` or `pull` from Lodash:

2

```
const _ = require('lodash');
_.without([1, 2, 3, 2], 2); // -> [1, 3]
```

edited Mar 18 '18 at 15:20

answered Jan 12 '16 at 22:39

sakovias

805 12 21

Please do not use the variant with `delete` - it makes a hole in the array as it does not re-index the elements after the deleted item.

1

```
> Array.prototype.remove=function(v){
...     delete this[this.indexOf(v)]
... };
[Function]
> var myarray=["3","24","55","2"];
undefined
> myarray.remove("55");
undefined
> myarray
[ '3', '24', , '2' ]
```

edited May 5 '14 at 16:56

mmohab

1,463 3 20 38

answered May 5 '14 at 16:10

Ilya Sher

361 3 3

I used the most voted option and created a function that would clean one array of words using another array of unwanted words:

1

```
function cleanArrayOfSpecificTerms(array,unwantedTermsArray) {
  $.each(unwantedTermsArray, function( index, value ) {
    var index = array.indexOf(value);
    if (index > -1) {
```

```
  return array;
}
```

To use, do the following:

```
var notInclude = ['Not','No','First','Last','Prior','Next', 'dogs','cats'];
var splitTerms = ["call", "log", "dogs", "cats", "topic", "change", "pricing"];

cleanArrayOfSpecificTerms(splitTerms,notInclude)
```

answered Feb 10 '15 at 16:06

maudulus

5,488 5 48 86

1

```
let arr = [5, 15, 25, 30, 35];
console.log(arr); //result [5, 15, 25, 30, 35]
let index = arr.indexOf(30);

if (index > -1) {
   arr.splice(index, 1);
}
console.log(arr); //result [5, 15, 25, 35]
```

answered Apr 18 '18 at 9:02

Asif vora

750 4 16

1

In a global function we can't pass a custom value directly but there are many way as below

```
var ary = ['three', 'seven', 'eleven'];
var index = ary.indexOf(item);//item: the value which you want to remove

//Method 1
ary.splice(index,1);
```

edited Aug 19 '18 at 18:54

answered Jun 28 '18 at 18:11

Srikrushna

1,221 15 27

delete shouldn't be used because it removes the value only but leave the space. – Vahid Akhtar Mar 17 at 14:57

0

```
var remove = function(array, value) {
    var index = null;

    while ((index = array.indexOf(value)) !== -1)
        array.splice(index, 1);

    return array;
};
```

answered Jan 13 '13 at 21:51

Alexander Abashkin

629 3 9 18

0

I tried using the function method from jbaron above but found that I needed to keep the original array intact for use later, and creating a new array like this:

```
var newArray = referenceArray;
```

apparently creates by reference instead of value because when I removed an element from newArray the referenceArray also had it removed. So I decided to create a new array each time like this:

```
function newArrRemoveItem(array, item, newArray){
    for(var i = 0; i < array.length; i++) {
        if(array[i]!=item){
            newArray.push(array[i]);
        }
```

Then I use it like this in another function:

```
var vesselID = record.get('VesselID');
var otherVessels = new Array();
newArrRemoveItem(vesselArr,vesselID,otherVessels);
```

Now the vesselArr remains intact while each time I execute the above code the otherVessels array includes all but the latest vesselID element.

answered May 29 '13 at 18:18

Robert
62 2 14

1 2 next

protected by j08691 Mar 20 '15 at 21:09

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?