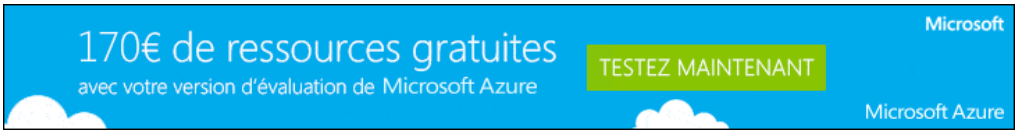
StackExchange 1 +52 69 4 help

Remove a particular element from an array in JavaScript?



I have an `array of integers`, which I'm using the `.push()` method to add to.

Is there a simple way to remove a specific element from an array? The equivalent of something like `array.remove(int);`

I have to use *core* JavaScript - *no* frameworks are allowed.

javascript arrays

edited Oct 8 '15 at 11:24
10011100010001
50 9

asked Apr 23 '11 at 22:17
Walker
14k 13 43 77

protected by Andrew Marshall Dec 15 '14 at 20:00

This question is protected to prevent "thanks!", "me too!", or spam answers by new users. To answer it, you must have earned at least 10 reputation on this site.

24 If you need to support **<IE9** (*sigh*) then check this SO question regarding `indexOf` in IE. – Scotty.NET Sep 13 '13 at 7:48

62 @Jonathon, my answer is for those (like me) who ended up on this page looking for a production solution, not specifically for Walker. `_.without([1, 2, 1, 0, 3, 1, 4], 0, 1); // => [2, 3, 4]` – zhon Sep 27 '13 at 20:42

5 See also: Remove item from array by value and Remove an array element by value in JavaScript – Bergi Aug 11 '14 at 16:48

|

36 Answers

1 2 next

First, find the `index` of the element you want to remove:

```
var array = [2, 5, 9];
var index = array.indexOf(5);
```

Note: browser support for indexOf is limited; it is not supported in Internet Explorer 7 and 8.

Then remove it with `splice`:

```
if (index > -1) {
    array.splice(index, 1);
}
```

The second parameter of `splice` is the number of elements to remove. Note that `splice` modifies the array in place and returns a new array containing the elements that have been removed.

edited Oct 8 '15 at 11:50
Mohammad Faisal
2,669 5 38 71

answered Apr 23 '11 at 22:23
Tom Wadley
41k 1 14 26

89 Good solution, but some browsers don't suppoer array.indexOf – Peter Olson Apr 23 '11 at 22:25

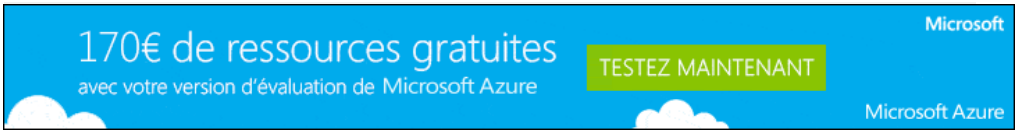
98 The second argument to the splice function is how many elements to remove. – Tom Wadley Apr 23 '11 at 22:25

44 array.indexOf is not supported in IE8 or earlier (see w3schools.com/jsref/jsref_indexof_array.asp). jQuery.inArray() works similarly though (api.jquery.com/jQuery.inArray) – Jon Onstott May 17 '13 at 17:15

27 Dangerous! If you are working with dynamic values (add and remove dynamic values from array) and value is not exist in array (the index become -1) the code above will remove the last element of the array. – Adrian P. Sep 6 '13 at 16:44

31 @GabrielFlorit No splice modifies the array. splice != slice. – Bleeding Fingers Apr 4 '14 at 18:24

|



I don't know how you are expecting `array.remove(int)` to behave. There are three possibilities I can think of that you might be wanting.

To remove an element of an array at an index `i`:

```
array.splice(i, 1);
```

If you want to remove every element with value `number` from the array:

```
for(var i = array.length - 1; i >= 0; i--) {
    if(array[i] === number) {
       array.splice(i, 1);
    }
}
```

If you just want to make the element at index `i` no longer exist, but you don't want the indexes of the other elements to change:

```
delete array[i];
```

edited May 23 '13 at 18:10

answered Apr 23 '11 at 22:20

Peter Olson
44.8k ● 23 ● 115 ● 175

113 `delete` is not the correct way to remove an element from an array! – Felix Kling Jan 27 '13 at 15:30

23 @FelixKling It depends, it works if you want to make it so that `array.hasOwnProperty(i)` returns `false` and have the element at that position return `undefined`. But I'll admit that that's not a very common thing to want to do. – Peter Olson Jan 27 '13 at 15:36

32 `delete` will not update the length of the array neither really erases the element, only replaces it with the special value `undefined`. – diosney Feb 17 '13 at 3:44

7 @diosney I don't know what you mean when you say it doesn't really erase the element. Further, it does more than simply replacing the value at that index with `undefined`: it removes both the index and the value from the array, i.e. after `delete array[0]`, `"0" in array` will return false. – Peter Olson Apr 15 '13 at 19:13

4 for(var i=array.length; i>=0; i--) should be for(var i=array.length-1; i>=0; i--) because indexing starts at 0 (there is no element at array[array.length]) – Bambax May 23 '13 at 18:04

|

Depends on whether you want to keep an empty spot or not.

If you do want an empty slot, delete is fine:

```
delete array[ index ];
```

If you don't, you should use the splice method:

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

And if you need the value of that item, you can just store the returned array's element:

```
var value = array.splice( index, 1 )[0];
```

In case you want to do it in some order, you can use `array.pop()` for the last one or `array.shift()` for the first one (and both return the value of the item too).

And if you don't know the index of the item, you can use `array.indexOf( item )` to get it (in a `if()` to get one item or in a `while()` to get all of them). `array.indexOf( item )` returns either

the index or -1 if not found.

edited Jul 10 '14 at 22:43
max taldykin
6,424 ● 3 ● 20 ● 50

answered Apr 23 '11 at 22:32
xavierm02
3,266 ● 1 ● 9 ● 20

31 It's worth noting that `var value` will not store the removed value but an array containing the removed value. – Jakub Jul 10 '13 at 9:55

8 `delete` is not the correct way to remove an element from an array!! – Progo Jul 4 '14 at 14:20

6 If you want to "empty a slot", use `array[index] = undefined;` . Using `delete` will destroy optimisation. – Bergi Aug 11 '14 at 16:40

1 @Jakub very good comment because to understand that I lost much time and thought my application code is somehow broken... – Pascal Dec 14 '14 at 19:14

A friend was having issues in Internet Explorer 8, and showed me what he did. I told him it was wrong, and he told me he got the answer here. The current top answer will not work in all browsers (Internet Explorer 8 for example), and it will only remove the first occurrence of the item.

Remove ALL instances from an array

```
function remove(arr, item) {
    for(var i = arr.length; i--;) {
        if(arr[i] === item) {
            arr.splice(i, 1);
        }
    }
}
```

It loops through the array backwards (since indices and length will change as items are removed) and removes the item if it's found. It works in all browsers.

edited Jul 11 '15 at 9:46
Peter Mortensen
9,254 ● 10 ● 66 ● 98

answered Aug 10 '13 at 19:21
Ben Lesh
70.3k ● 32 ● 197 ● 200

7 @sroes it should not be because the loop starts at `i = arr.length -1` or `i--` making it same as the max index. `arr.length` is just an initial value for `i` . `i--` will always be **truthy** (and reducing by 1 at each loop op) until it equals `0` (a falsy value) and the loop will then stop. – gabeno Jan 29 '14 at 20:06

1 Oh, now I see. The lack of a third statement threw me off. – sroes Jan 29 '14 at 20:35

1 Second function is rather inefficient. On every iteration "indexOf" will start search from beginning of array. – Amber de Black Feb 25 '15 at 13:32

|

Remarks

- This function removes every occurence of specified value from array.
- Function name have "stackoverflow_" prefix to prevent name collision. If you accepts the risk of name collision, you can remove that prefix.
- There are described 4 versions of this function for different cases.

Option #1 Extending "Array.prototype" with "Object.defineProperty" function

Compatible browsers: Internet Explorer 9+, Firefox 4+, Chrome 5+, Safari 5+, and Opera 12+

Extend the Array prototype by using "Object.defineProperty" function.

This approach will not cause problems with enumeration, because we marked "enumerable" as "false".

Be sure that your browser supports "Object.defineProperty" function. Here is the compatibility table:

http://kangax.github.io/es5-compat-table/#Object.defineProperty

Extension code:

```
// Extending Array prototype with new function,
// if that function is already defined in "Array.prototype",
// then "Object.defineProperty" will throw an exception
Object.defineProperty(Array.prototype, "stackoverflow_remove", {
    // Specify "enumerable" as "false" to prevent function enumeration
    enumerable: false,

    /**
    * Removes all occurence of specified item from array
    * @this Array
    * @param itemToRemove Item to remove from array
    * @returns {Number} Count of removed items
    */
    value: function (itemToRemove) {
        // Count of removed items
        var removeCounter = 0;

        // Iterate every array item
        for (var index = 0; index < this.length; index++) {
            // If current array item equals itemToRemove then
            if (this[index] === itemToRemove) {
                // Remove array item at current index
                this.splice(index, 1);

                // Increment count of removed items
                removeCounter++;

                // Decrement index to iterate current position
                // one more time, because we just removed item
                // that occupies it, and next item took it place
                index--;
            }
        }

        // Return count of removed items
        return removeCounter;
    }
});
```

Usage code #1:

```
var arr = [1, 2, 3, 2, 2, 2];

var itemsRemoved = arr.stackoverflow_remove(2);

console.log(itemsRemoved);
// 4

console.log(arr);
// [1, 3]
```

Usage code #2:

```
var arr = ["tree", "bird", "car", "bird", "bird"];

var itemsRemoved = arr.stackoverflow_remove("bird");

console.log(itemsRemoved);
// 3

console.log(arr);
// ["tree", "car"]
```

Option #2 Defining global function. For old browsers which not support prototype extending with "Object.defineProperty"

If you want to use this function without "Object.defineProperty", you can define it as a global scope function.

Extension code:

```
/**
* Removes all occurence of specified item from array
* @param array Array
* @param itemToRemove Item to remove from array
* @returns {Number} Count of removed items
*/
function stackoverflow_removeArrayItem(array, itemToRemove) {
    // Count of removed items
    var removeCounter = 0;

    // Iterate every array item
    for (var index = 0; index < array.length; index++) {
        // If current array item equals itemToRemove then
        if (array[index] === itemToRemove) {
            // Remove array item at current index
```

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

            // Increment count of removed items
            removeCounter++;

            // Decrement index to iterate current position
            // one more time, because we just removed item
            // that occupies it, and next item took it place
            index--;
        }
    }

    // Return count of removed items
    return removeCounter;
}
```

Usage code:

```
var arr = ["tree", "bird", "car", "bird", "bird"];

var itemsRemoved = stackoverflow_removeArrayItem(arr, "bird");

console.log(itemsRemoved);
// 3

console.log(arr);
// ["tree", "car"]
```

Option #3 For high performance

This code uses a "filter" function and it works about 50 times faster than previous options, but this approach creates new array.

Extension code:

```
// Extending Array prototype with new function,
// if that function is already defined in "Array.prototype",
// then "Object.defineProperty" will throw an exception
Object.defineProperty(Array.prototype, "stackoverflow_filterValue", {
    // Specify "enumerable" as "false" to prevent function enumeration
    enumerable: false,

    /**
    * Create new array where specified item is removed
    * @this Array
    * @param itemToRemove Item to remove from array
    * @returns {Number} Count of removed items
    */
    value: function (itemToRemove) {
        var filteredArray = this.filter(function(item){
            return item !== itemToRemove;
        });

        return filteredArray;
    }
});
```

Usage code:

```
var arr = [1, 2, 3, 2, 2, 2];

// PAY ATTENTION.
// Original array stay unchanged.
var filteredArray = arr.stackoverflow_filterValue(2);

console.log(filteredArray);
// [1, 3]
```

Option #4 ECMAScript 2015 way (if your browser support modern JavaScript or you use Babel.js)

Using new version of JavaScript we need no custom functions to remove array items. Using only filter(...) function and arrow function we got very tiny code:

```
let value = 3;

let arr = [1, 2, 3, 4, 5, 3];

arr = arr.filter(item => item !== value);

console.log(arr);
// [ 1, 2, 4, 5 ]
```

edited Oct 9 '15 at 14:27 answered Dec 19 '13 at 19:54

Amber de Black
1,078 5 14

4 The risk you run when you add methods to built-in prototypes is that a future version of JavaScript will implement a method with the same name, and then your code will break any scripts that depend on the built-in behavior, or your scripts will break because your API may not be compatible with the built-in API. – Eric Elliott Nov 18 '14 at 18:16

1 Amber, 1) Nice idea, but few people actually do that. (See your answer for proof). 2) Even if people did that, what's to prevent different authors from choosing the same prefix? For much better solutions, see jQuery, Underscore, Lodash, Highland.js, etc...: `$('#el').newMethod()`, `_(myArray).filterValue(2)`, etc... – Eric Elliott Nov 26 '14 at 3:46

3 The shit risk is this method freezes datepicker (jquery plugin) because somewhere is does use the same method overwritten. Thanks the devil i could find the bug with a nice debugger. I had to rename the method "Object.defineProperty(Array.prototype, "_remove") – Ismael Dec 11 '14 at 16:12

|

The easiest way:

```
array.splice( array.indexOf(item), 1 );
```

answered Jul 21 '14 at 14:03

Nanego
1,047 8 20

2 I love one liners :) – Cyril N. Feb 5 at 14:44

Be careful when you use delete for an array. It is good for deleting attributes of objects but not so good for arrays. It is better to use `splice` for arrays.

Keep in mind that when you use `delete` for an array you could get wrong results for `anArray.length`. In other words, `delete` would remove the element but wouldn't update the value of length property.

You can also expect to have holes in index numbers after using delete, e.g. you could end up with having indexes 1,3,4,8,9,11 and length as it was before using delete. In that case, all indexed `for` loops would crash, since indexes are no longer sequential.

If you are forced to use `delete` for some reason, then you should use `for each` loops when you need to loop through arrays.

edited Dec 29 '15 at 9:33

answered Dec 21 '12 at 11:32

Sasa
733 8 21

```
Array.prototype.remByVal = function(val) {
    for (var i = 0; i < this.length; i++) {
        if (this[i] === val) {
            this.splice(i, 1);
            i--;
        }
    }
    return this;
}
//Call like
[1, 2, 3, 4].remByVal(3);
```

answered Apr 23 '11 at 22:20

Zirak
16.7k 6 47 74

9 I'm not a big fan of this approach. If you end up using different libraries or frameworks, they can end up conflicting with each other. – Charlie Kilian Apr 23 '11 at 22:30

6 Bad idea, see this post: stackoverflow.com/questions/948358/array-prototype-problem – MMeah Jul 9 '12 at 22:10

8 If you're doing a `for in` on an array, you already have a problem. – Zirak May 14 '14 at 13:01

|

There is no need to use `indexOf` or `splice`. However, it performs better if you only want to remove one occurrence of an element.

Find and move (move):

```
function move(arr, val) {
  var j = 0;
  for (var i = 0, l = arr.length; i < l; i++) {
    if (arr[i] !== val) {
      arr[j++] = arr[i];
    }
  }
  arr.length = j;
}
```

Use `indexOf` and `splice` (indexof):

```
function indexof(arr, val) {
  var i;
  while ((i = arr.indexOf(val)) != -1) {
    arr.splice(i, 1);
  }
}
```

Use only `splice` (splice):

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

Run-times on nodejs for array with 1000 elements (average over 10000 runs):

indexof is approximately 10x slower than *move*. Even if improved by removing the call to `indexOf` in *splice* it performs much worse than *move*.

```
Remove all occurrences:
    move 0.0048 ms
    indexof 0.0463 ms
    splice 0.0359 ms

Remove first occurrence:
    move_one 0.0041 ms
    indexof_one 0.0021 ms
```

edited Sep 11 '15 at 12:47

answered Sep 19 '13 at 1:53

slosd
1,176 10 16

|

John Resig posted a good implementation:

```
// Array Remove - By John Resig (MIT Licensed)
Array.prototype.remove = function(from, to) {
  var rest = this.slice((to || from) + 1 || this.length);
  this.length = from < 0 ? this.length + from : from;
  return this.push.apply(this, rest);
};
```

If you don't want to extend a global object, you can do something like the following, instead:

```
// Array Remove - By John Resig (MIT Licensed)
Array.remove = function(array, from, to) {
    var rest = array.slice((to || from) + 1 || array.length);
    array.length = from < 0 ? array.length + from : from;
    return array.push.apply(array, rest);
};
```

But the main reason I am posting this is to warn users against the alternative implementation suggested in the comments on that page (Dec 14, 2007):

```
Array.prototype.remove = function(from, to){
  this.splice(from, (to=[0,from||1,++to-from][arguments.length])<0?this.length+to:to);
  return this.length;
};
```

It seems to work well at first, but through a painful process I discovered it fails when trying to remove the second to last element in an array. For example, if you have a 10-element array

and you try to remove the 9th element with this:

```
myArray.remove(8);
```

You end up with an 8-element array. Don't know why but I confirmed John's original implementation doesn't have this problem.

edited Feb 16 '15 at 12:51
magiccrafter
329 3 9

answered Aug 30 '13 at 19:07
Roger
457 7 16

|

You can do it easily with filter method:

```
function remove(arrOriginal, elementToRemove){
    return arrOriginal.filter(function(el){return el !== elementToRemove});
}
console.log( remove([1, 2, 1, 0, 3, 1, 4], 1) );
```

This removes all elements from the array and also works faster then combination of slice and indexOf

answered Feb 10 '14 at 22:06
Salvador Dali
37.5k 29 174 248

2 Also note, Array.prototype.filter is ECMAScript 5.1 (No IE8) – Montana Harkin May 23 '14 at 20:35

|

Underscore.js can be used to solve issues with multiple browsers. It uses in-build browser methods if present. If they are absent like in the case of older IE it uses its own custom methods.

Simple example to remove elements from array (from the website) -

```
_.without([1, 2, 1, 0, 3, 1, 4], 0, 1); // => [2, 3, 4]
```

answered May 30 '14 at 9:57
vatsal
1,551 9 14

If you want a new array with the deleted positions removed, you can always delete the specific element and filter out the array. It might need an extension of the array object for browsers that don't implement the filter method but in the long term its easier since all you do is this:

```
var my_array = [1,2,3,4,5,6];
delete my_array[4];
console.log(my_array.filter(function(a){return typeof a !== 'undefined';}));
```

Should display `[1, 2, 3, 4, 6]`

edited Sep 5 '14 at 7:59

answered Oct 18 '12 at 10:13
Loupax
1,334 18 39

Check out this code. It works in every **major browser**.

```
remove_item = function (arr, value) {
    var b = '';
    for (b in arr) {
        if (arr[b] === value) {
            arr.splice(b, 1);
            break;
        }
    }
    return arr;
}
```

Call this function

```
remove_item(array,value);
```

edited May 24 '14 at 7:02
Cupcake
56.5k • 17 • 118 • 151

answered Apr 2 '13 at 10:56
Farhad
1,636 • 20 • 22

|

I'm pretty new to JavaScript and needed this functionality. I merely wrote this:

```
function removeFromArray(array, item, index) {
  while((index = array.indexOf(item)) > -1) {
    array.splice(index, 1);
  }
}
```

Then when I want to use it:

```
//Set-up some dummy data
var dummyObj = {name:"meow"};
var dummyArray = [dummyObj, "item1", "item1", "item2"];

//Remove the dummy data
removeFromArray(dummyArray, dummyObj);
removeFromArray(dummyArray, "item2");
```

Output - As expected. ["item1", "item1"]

You may have different needs than I, so you can easily modify it to suit them. I hope this helps someone.

edited Dec 30 '14 at 16:17
yckart
12.2k • 4 • 54 • 77

answered Jan 16 '14 at 11:27
sofiax
151 • 1 • 3

I know too old to reply, but I want to add my functions that take a predicate instead of a value.

Definition

```
var ArrayHelper = {

    /**
     * Remove the first occurrence
     *
     * @param Array
     * @param function
     * @returns the removed item
     *
     */
    remove: function(array, predict) {
        for (var i = 0; i < array.length; i++) {
            if (predict(array[i])) {
                return array.splice(i, 1);
            }
        }
    },

    /**
     * Remove all occurrences
     *
     * @param Array
     * @param function
     * @returns the removed items
     *
     */
    removeAll: function(array, predict) {
        var removed = [];
        for (var i = 0; i < array.length;) {
            if (predict(array[i])) {
                removed.push(array.splice(i, 1));
            }else{
                i++;
            }
        }

        return removed;
    }
};
```

Usage

```
ArrayHelper.remove(myArray, function(row) { return row.id === 5 });
```

```
ArrayHelper.removeAll(myArray, function(row) { return row.id > 3 && row.id < 15});
```

Hope this helps

edited Dec 17 '15 at 12:07

answered May 2 '14 at 12:00

Ahmad
7,085 ● 2 ● 19 ● 31

|

You can use lodash _.pull (mutate array), _.pullAt (mutate array) or _.without (does't mutate array),

```
var array1 = ['a', 'b', 'c', 'd']
_.pull(array1, 'c')
console.log(array1) // ['a', 'b', 'd']

var array2 = ['e', 'f', 'g', 'h']
_.pullAt(array2, 0)
console.log(array2) // ['f', 'g', 'h']

var array3 = ['i', 'j', 'k', 'l']
var newArray = _.without(array3, 'i') // ['j', 'k', 'l']
console.log(array3) // ['i', 'j', 'k', 'l']
```

edited Aug 25 '15 at 21:19

answered Aug 25 '15 at 19:34

Chun Yang
693 ● 7 ● 12

1 @some-non-descript-user You are right. But a lot of users like me come here looking for a general answer not just for the OP only. – Chun Yang Oct 1 '15 at 3:38

|

Use jQuery's InArray:

```
A = [1, 2, 3, 4, 5, 6];
A.splice($.inArray(3, A), 1);
//It will return A=[1, 2, 4, 5, 6]`
```

Note: inArray will return -1, if the element was not found.

edited Jul 11 '15 at 9:48

Peter Mortensen
9,254 ● 10 ● 66 ● 98

answered Sep 17 '14 at 10:14

Do Hoa Vinh
129 ● 1 ● 4

4 but OP said: "good ol' fashioned JavaScript - no frameworks allowed" – CS♀ Dec 12 '14 at 18:51

I know there are a lot of answers already, but many of them seem to over complicate the problem. Here is a simple, recursive way of removing all instances of a key - calls self until index isn't found. Yes, it only works in browsers with `indexOf`, but it's simple and can be easily polyfilled.

Stand-alone function

```
function removeAll(array, key){
    var index = array.indexOf(key);

    if(index === -1) return;

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

Prototype method

```
Array.prototype.removeAll = function(key){
    var index = this.indexOf(key);

    if(index === -1) return;

    this.splice(index, 1);
    this.removeAll(key);
}
```

edited Feb 19 '14 at 22:20

answered Feb 3 '14 at 15:41
slccsoccer28
422 • 4 • 11

|

If you have complex objects in the array you can use filters? In situations where $.inArray or array.splice is not as easy to use. Especially if the objects are perhaps shallow in the array.

E.g. if you have an object with an Id field and you want the object removed from an array:

```
this.array = this.array.filter(function(element, i) {
    return element.id !== idToRemove;
});
```

edited Nov 10 '15 at 4:20
Anik Islam Abhi
16.2k • 7 • 18 • 40

answered Apr 9 '15 at 10:00
flurdy
1,674 • 11 • 18

|

This gist here will solve your problem, and also deletes all occurrences of the argument instead of just 1 (or a specified value).

```
Array.prototype.destroy = function(obj){
    // Return null if no objects were found and removed
    var destroyed = null;

    for(var i = 0; i < this.length; i++){

        // Use while-loop to find adjacent equal objects
        while(this[i] === obj){

            // Remove this[i] and store it within destroyed
            destroyed = this.splice(i, 1)[0];
        }
    }

    return destroyed;
}
```

Usage:

```
var x = [1, 2, 3, 3, true, false, undefined, false];

x.destroy(3);         // => 3
x.destroy(false);     // => false
x;                    // => [1, 2, true, undefined]

x.destroy(true);      // => true
x.destroy(undefined); // => undefined
x;                    // => [1, 2]

x.destroy(3);         // => null
x;                    // => [1, 2]
```

edited May 13 '13 at 12:58

answered Mar 13 '13 at 9:28
zykadelic
486 • 6 • 14

1 This is buggy on unsorted lists. `[1,2,3,3,2,1].destroy(1)` results in `[3,3,2,1]` plnkr.co/edit/p8QhmOfgl9AzlBWjLLTp?p=preview – Walter Stabosz May 2 '13 at 20:45

|

I also ran in the situation where I had to remove an element from `Array`. `.indexOf` was not working in `IE*` so sharing my working `jQuery.inArray()` solution.

```
var index = jQuery.inArray(val,arr);
if (index > -1) {
    arr.splice(index, 1);
    //console.log(arr);
}
```

answered Oct 8 '13 at 10:09
NullPointer
1,575 • 1 • 11 • 35

|

In CoffeeScript:

```
my_array.splice(idx, 1) for ele, idx in my_array when ele is this_value
```

edited Sep 5 '14 at 7:21

answered Jan 30 '14 at 4:27

Nigel Sheridan-Smith
325 • 2 • 6

```
Array.prototype.removeItem = function(a) {
          for (i = 0; i < this.length; i++) {
              if (this[i] == a) {
                  for (i2 = i; i2 < this.length - 1; i2++) {
                      this[i2] = this[i2 + 1];
                  }
                  this.length = this.length - 1
                  return;
              }
          }
      }

  var recentMovies = ['Iron Man', 'Batman', 'Superman', 'Spiderman'];
  recentMovies.removeItem('Superman');
```

answered Sep 26 '13 at 0:12

Don Vincent Preziosi
29 • 3

Create new array:

```
var my_array = new Array();
```

Add elements to this array:

```
my_array.push("element1");
```

The function indexOf (Returns index or -1 when not found) :

```
var indexOf = function(needle)
{
    if(typeof Array.prototype.indexOf === 'function') // newer browsers
    {
        indexOf = Array.prototype.indexOf;
    }
    else // older browsers
    {
        indexOf = function(needle)
        {
            var index = -1;

            for(var i = 0; i < this.length; i++)
            {
                if(this[i] === needle)
                {
                    index = i;
                    break;
                }
            }
            return index;
        };
    }

    return indexOf.call(this, needle);
};
```

Check index of this element (tested with firefox and IE8+):

```
var index = indexOf.call(my_array, "element1");
```

Remove 1 element located at index from the array

```
my_array.splice(index, 1);
```

answered Aug 7 '13 at 12:57

Enrico
187 • 4 • 18

You can do a backward loop to make sure not to screw up the indexes, if there are multiple instances of the element.

```
var myElement = "chocolate";
var myArray = ['chocolate', 'poptart', 'poptart', 'poptart', 'chocolate', 'poptart',
'poptart', 'chocolate'];

/* Important code */
for (var i = myArray.length - 1; i >= 0; i--) {
    if (myArray[i] == myElement) myArray.splice(i, 1);
}
```

Live Demo

answered Aug 12 '13 at 17:56

Jeff Noel
3,988 ● 2 ● 15 ● 44

There are many fantastic answers here, but for me, what worked most simply wasn't removing my element from the array completely but simply setting the value of it to null. This works for most cases I have, and is a good solution since I will be using the variable later and don't want it gone, just empty for now. Also, this approach is completely cross-browser compatible.

```
array.key = null;
```

answered Jan 7 '15 at 19:53

rncrtr
1,842 ● 1 ● 8 ● 24

Removing the value with index and splice!

```
function removeArrValue(arr,value) {
    var index = arr.indexOf(value);
    if (index > -1) {
        arr.splice(index, 1);
    }
    return arr;
}
```

edited Jul 11 '15 at 9:49

Peter Mortensen
9,254 ● 10 ● 66 ● 98

answered Oct 22 '14 at 14:10

Nejc Lepen
39 ● 3

10 Your 2 last comments were just rewriting an accepted answer... Please answer a solved problem only if you have more information to provide than the accepted one. If not, just upvote the accepted answer. – Miam84 Oct 22 '14 at 14:43

You can iterate over each `array`-item and `splice` it if it exist in your `array`.

```
function destroy(arr, val) {
    for (var i = 0; i < arr.length; i++) if (arr[i] === val) arr.splice(i, 1);
    return arr;
}
```

answered May 13 '13 at 12:22

yckart
12.2k ● 4 ● 54 ● 77

|

I like this version of splice, removing an element by its value using `$.inArray`:

```
$(document).ready(function(){
    var arr = ["C#","Ruby","PHP","C","C++"];
    var itemtoRemove = "PHP";
    arr.splice($.inArray(itemtoRemove, arr),1);
});
```

edited Jun 22 '14 at 22:03

Peter Mortensen
9,254 ● 10 ● 66 ● 98

answered Mar 20 '14 at 9:56

mboeckle
434 ● 7 ● 16

1 yes correct, you should know which element you want to remove like in the other examples. – mboeckle
May 1 '14 at 17:00

|

1 2 next

Answer This Question