How do I remove objects from a javascript associative array?

Asked 10 years, 10 months ago Active 3 months ago Viewed 523k times

583

69

Suppose I have this code:

```
var myArray = new Object();
myArray["firstname"] = "Bob";
myArray["lastname"] = "Smith";
myArray["age"] = 25;
```

Now if I wanted to remove "lastname"?....is there some equivalent of `myArray["lastname"].remove()` ?

(I need the element gone because the number of elements is important and I want to keep things clean.)

javascript arrays associative-array associative

edited Oct 7 '13 at 10:39
djot
2,762 3 15 26

asked Dec 6 '08 at 6:44
Andrew

26 A tip: don't get arrays and maps confused. Some languages, like php, have a single object for both. Though you used the right type here (new Object()) you named it myArray, it's just a matter of standards for a langugage. – Juan Mendes Apr 27 '10 at 17:35

Don't forget that JavaScript is type-less and everything is an object. See Saul's answer below. – meshfields Jun 15 '12 at 9:24

4 @StephanKristyn - to be precise, JS has types but in a dynamic and weak way. For example, while its variables indeed are typeless, their values are not. That is the **dynamic** part. **Weak** denotes that operations between different value types are not strictly defined and rely on behind-the-scenes conversions; for example `"Test" + {};` is a perfectly valid JS statement. – Saul Aug 27 '12 at 8:46

17 Answers

1083

```
delete myArray["lastname"];
```

✓

EDIT:

In some JavaScript engine, the delete keyword might hurt performance as it will undo compile / JIT optimization.

http://www.html5rocks.com/en/tutorials/speed/v8/ http://www.smashingmagazine.com/2012/11/writing-fast-memory-efficient-javascript/

edited Aug 19 '15 at 5:49

answered Dec 6 '08 at 6:46
Dennis C
20.6k 12 63 95

17 This will cause problems if used on an Array object instance to remove an existing element, e.g. `delete myArray[0]` . See stackoverflow.com/a/9973592/426379 and Deleting array elements – Saul Apr 2 '12 at 9:43

4 What problems will be caused? – Gottox Apr 2 '12 at 10:01

24 @Gottox - The `length` property of an Array object remains unchanged. – Saul Apr 2 '12 at 10:53

12 @Saul: there *would* be problems if `myArray` was really used as an array - but it is not (`myArray` is unfortunate name), it is an object. So in this case `delete` is OK. Note that even if it was created as `new Array()` and used as associative array it would still be OK. Your warning is still something to be aware of if one is using real arrays though. – johndodo Apr 4 '12 at 7:47

2 @johndodo - True. That is why I started my initial comment with *This will cause problems **if** used on an **Array** object instance*. I nevertheless prefer an approach which performs correctly in all cases, see my answer below. – Saul Apr 4 '12 at 9:19

75

All objects in JavaScript are implemented as hashtables/associative arrays. So, the following are the equivalent:

```
alert(myObj["SomeProperty"]);
alert(myObj.SomeProperty);
```

And, as already indicated, you "remove" a property from an object via the `delete` keyword, which you can use in two ways:

```
delete myObj["SomeProperty"];
delete myObj.SomeProperty;
```

answered Dec 6 '08 at 7:28
Jason Bunting
50.8k 12 93 92

8 should be noted that the dot notation doesn't work if the property isn't a simple term. i.e. `myObj['some;property']` works, but `myObj.some;property` wouldn't (for obvious reasons). Also it might not be obvious that you can use a variable in the bracket notation, i.e. `var x = 'SomeProperty'; alert(myObj[x])` – Kip Apr 27 '11 at 3:48

2 "All objects in JavaScript are implemented as hashtables/associative arrays. " - false. V8 prefers to store an object as a hidden class + densely packed fields. Only if you do weird stuff to them (such as removing fields) it gives up and uses a hash map behind the scenes. – John Dvorak Aug 19 '15 at 6:08

4 @JanDvorak - hey, you recognize when this answer was originally written, yeah? That description was and still is sufficient for most purposes. That said, I understand being tediously pedantic. :) – Jason Bunting Nov 10 '15 at 20:23

41

None of the previous answers address the fact that Javascript does not have associative arrays to begin with - there is no `array` type as such, see `typeof` .

What Javascript has, are object instances with dynamic properties. When properties are confused with elements of an Array object instance then Bad Things™ are bound to happen:

Problem

```
var elements = new Array()

elements.push(document.getElementsByTagName("head")[0])
elements.push(document.getElementsByTagName("title")[0])
elements["prop"] = document.getElementsByTagName("body")[0]

console.log("number of elements: ", elements.length)   // returns 2
delete elements[1]
console.log("number of elements: ", elements.length)   // returns 2 (?!)

for (var i = 0; i < elements.length; i++)
{
   // uh-oh... throws a TypeError when i == 1
   elements[i].onmouseover = function () { window.alert("Over It.")}
   console.log("success at index: ", i)
}
```

To have a universal removal function that does not blow up on you, use:

```
Object.prototype.removeItem = function (key) {
   if (!this.hasOwnProperty(key))
      return
   if (isNaN(parseInt(key)) || !(this instanceof Array))
      delete this[key]
   else
      this.splice(key, 1)
};

//
// Code sample.
//
var elements = new Array()

elements.push(document.getElementsByTagName("head")[0])
elements.push(document.getElementsByTagName("title")[0])
elements["prop"] = document.getElementsByTagName("body")[0]

console.log(elements.length)                          // returns 2
elements.removeItem("prop")
elements.removeItem(0)
console.log(elements.hasOwnProperty("prop"))          // returns false as it should
console.log(elements.length)                          // returns 1 as it should
```

edited Mar 22 '18 at 15:50

answered Apr 2 '12 at 9:07
Saul
15.5k 6 54 85

7 This solution has two issues: it hides the fact that arrays and objects are entirely different beasts in JS (you know it, but apparently OP doesn't) and it uses prototypes. OP would be better off if he learned about arrays and objects (and would name his variables accordingly) - trying to hide the differences between the two will only get him in more trouble. IMHO of course. – johndodo Apr 4 '12 at 13:38

1 @johndodo - all `Array` s in JS are objects, try `typeof new Array();` or `typeof []` to verify. `Array` is simply a certain kind of an object and not at all a "different beast". In JS, objects are distinguished by their constructor name and prototype chain, see Prototype-based programming. – Saul Apr 5 '12 at 6:53

8 You are missing the point. I know that arrays are objects too, but that doesn't mean it is wise to use them as such. Programmer should decide if he wants to use something as array (with push, pop, [],...) or as object/"associative array". Mix and match is not a good recipe, precisely because of the problems your solution is trying to hide. If you decide in advance which design pattern to use (array or object) there will be no such problems. – johndodo Apr 5 '12 at 8:06

28

That only removes deletes the object but still keeps the array length same.

To remove you need to do something like:

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

edited Apr 5 '12 at 8:06
Alex
28.9k 9 66 124

answered Sep 13 '10 at 3:41
Bipin
345 3 2

10 Indeed, but in this case an array is not being used, just a plain old object, thus it has no length or splice method. – MooGoo Sep 13 '10 at 3:48

this is 99.99% of the time the right call – Andreas Panagiotidis Jan 24 '18 at 14:53

2 @Andreaa Panagiotidis Except when we're not talking about Arrays, in which case it's wrong 100% of the time ☺ – Drenai May 31 '18 at 1:32

15

While the accepted answer is correct, it is missing the explanation why it works.

First of all, your code should reflect the fact that this is **NOT** an array:

```
var myObject = new Object();
myObject["firstname"] = "Bob";
myObject["lastname"] = "Smith";
myObject["age"] = 25;
```

Note that all objects (including `Array` s) can be used this way. However, do not expect for standard JS array functions (pop, push,...) to work on objects!

As said in accepted answer, you can then use `delete` to remove the entries from objects:

```
delete myObject["lastname"]
```

You should decide which route you wish to take - either use objects (associative arrays / dictionaries) or use arrays (maps). Never mix the two of them.

edited Apr 5 '12 at 8:18

answered Apr 5 '12 at 8:04
johndodo
8,463 9 63 96

5 Very good answer. I would only advise anyone reading this that Arrays in javascript should not be abstracted as 'maps', but rather 'lists'. That's because you should not try to have control over the index of the elements when using arrays. If you try that...well, just don't :D – rodolfo42 May 4 '14 at 3:13

7

Use method `splice` to completely remove item from an object array:

```
Object.prototype.removeItem = function (key, value) {
    if (value == undefined)
        return;

    for (var i in this) {
        if (this[i][key] == value) {
            this.splice(i, 1);
        }
    }
};

var collection = [
    { id: "5f299a5d-7793-47be-a827-bca227dbef95", title: "one" },
    { id: "87353080-8f49-46b9-9281-162a41ddb8df", title: "two" },
    { id: "a1af832c-9028-4690-9793-d623ecc75a95", title: "three" }
];

collection.removeItem("id", "87353080-8f49-46b9-9281-162a41ddb8df");
```

edited Jul 23 '15 at 5:24

answered Jul 23 '15 at 5:06
HarpyWar
115 1 5

1 this is a more generic solution, can be added to your js file and the method will be available to all arrays, not just one array. – Hussain Jun 29 '17 at 12:24

5

You are using Object, you are not having an associative array to begin with. With an associative array, adding and removing items goes like this:

```
Array.prototype.contains = function(obj)
{
    var i = this.length;
    while (i--)
    {
        if (this[i] === obj)
        {
            return true;
        }
    }
    return false;
}


Array.prototype.add = function(key, value)
{
    if(this.contains(key))
        this[key] = value;
    else
    {
        this.push(key);
        this[key] = value;
    }
}


Array.prototype.remove = function(key)
{
    for(var i = 0; i < this.length; ++i)
    {
        if(this[i] == key)
        {
            this.splice(i, 1);
            return;
        }
    }
}



// Read a page's GET URL variables and return them as an associative array
```

```
        var hashes = window.location.href.slice(window.location.href.indexOf('?') +
1).split('&');

        for(var i = 0; i < hashes.length; i++)
        {
            hash = hashes[i].split('=');
            vars.push(hash[0]);
            vars[hash[0]] = hash[1];
        }

        return vars;
    }



    function ForwardAndHideVariables() {
        var dictParameters = getUrlVars();

        dictParameters.add("mno", "pqr");
        dictParameters.add("mno", "stfu");

        dictParameters.remove("mno");



        for(var i = 0; i < dictParameters.length; i++)
        {
            var key = dictParameters[i];
            var value = dictParameters[key];
            alert(key + "=" + value);
        }
        // And now forward with HTTP-POST
        aa_post_to_url("Default.aspx", dictParameters);
    }


    function aa_post_to_url(path, params, method) {
        method = method || "post";

        var form = document.createElement("form");

        //move the submit function to another variable
        //so that it doesn't get written over if a parameter name is 'submit'
        form._submit_function_ = form.submit;

        form.setAttribute("method", method);
        {
            var key = params[i];

            var hiddenField = document.createElement("input");
            hiddenField.setAttribute("type", "hidden");
            hiddenField.setAttribute("name", key);
            hiddenField.setAttribute("value", params[key]);

            form.appendChild(hiddenField);
        }

        document.body.appendChild(form);
        form._submit_function_(); //call the renamed function
    }
```

edited Oct 17 '13 at 18:22

answered Jun 1 '11 at 8:57
Stefan Steiger
49.5k 57 290 376

5

As other answers have noted, what you are using is not a Javascript array, but a Javascript object, which works almost like an associative array in other languages except that all keys are converted to strings. The new Map stores keys as their original type.

If you had an array and not an object, you could use the array's .filter function, to return a new array without the item you want removed:

```
var myArray = ['Bob', 'Smith', 25];
myArray = myArray.filter(function(item) {
    return item !== 'Smith';
});
```

If you have an older browser and jQuery, jQuery has a `$.grep` method that works similarly:

```
myArray = $.grep(myArray, function(item) {
    return item !== 'Smith';
});
```

edited Mar 29 '16 at 18:32
Zev Spitz

answered Jun 21 '12 at 15:24
Dzeimsas Zvirblis

perfect explanation. I used filter to achieve the desired result. Would you explain how the return item works to remove the object from the array. I'm assuming it returns the array as long as it doesn't include the string you included. – Edward Jan 19 '18 at 16:05

5

There is an elegant way in Airbnb Style Guide to do this (ES7):

```
const myObject = {
  a: 1,
  b: 2,
  c: 3
};
const { a, ...noA } = myObject;
console.log(noA); // => { b: 2, c: 3 }
```

Copyright: https://codeburst.io/use-es2015-object-rest-operator-to-omit-properties-38a3ecffe90

answered Oct 5 '18 at 11:51
Pavel Druzhinin
108 1 7

3

If for whatever reason the delete key is not working (like it wasnt working for me)

You can splice it out and then filter the undefined values

```
// to cut out one element via arr.splice(indexToRemove, numberToRemove);
array.splice(key, 1)
array.filter(function(n){return n});
```

Dont try and chain them since splice returns removed elements;

edited May 20 '13 at 2:19

answered May 20 '13 at 2:12
Leon
2,855 26 33

3

myArray["lastname"] = undefined;

answered Jan 4 '15 at 17:59
Amytis
41 4

This could be useful in cases where one isn't sure whether the key exists in the dictionary, but wants to sanitize it in case it does. Correct me if I'm wrong Amytis. – Hassan Baig Jun 12 '18 at 21:27

2

Its very straight forward if you have underscore.js dependency in your project -

```
_.omit(myArray, "lastname")
```

answered May 29 '14 at 19:23
vatsal
2,829 1 14 19

2

We can use it as a function too. Angular throws some error if used as a prototype. Thanks @HarpyWar. It helped me solve a problem.

```
var removeItem = function (object, key, value) {
    if (value == undefined)
        return;

    for (var i in object) {
        if (object[i][key] == value) {
            object.splice(i, 1);
        }
    }
};

var collection = [
    { id: "5f299a5d-7793-47be-a827-bca227dbef95", title: "one" },
    { id: "87353080-8f49-46b9-9281-162a41ddb8df", title: "two" },
removeItem(collection, "id", "87353080-8f49-46b9-9281-162a41ddb8df");
```

answered Jan 7 '16 at 18:47

Omkar Kamale

21 1

1

By using the `"delete"` keyword, it will delete the array element from array in javascript.

For example,

Consider following statements.

```
var arrayElementToDelete = new Object();

arrayElementToDelete["id"]           = "XERTYB00G1";
arrayElementToDelete["first_name"]   = "Employee_one";
arrayElementToDelete["status"]       = "Active";

delete arrayElementToDelete["status"];
```

Last line of the code will remove the array element who's key is "status" from the array.

answered Aug 6 '14 at 5:06

Ravindra Miyani

167 4 13

0

```
var myArray = newmyArray = new Object();
myArray["firstname"] = "Bob";
myArray["lastname"] = "Smith";
myArray["age"] = 25;

var s = JSON.stringify(myArray);

s.replace(/"lastname[^,}]+,/g,'');
```

Without looping/iterates we get the same result

edited Jun 15 '17 at 8:36
WhatsThePoint
2,545 6 23 39

answered Jun 14 '17 at 7:25
Lalith kumar
11 4

0

For "Arrays":

If you know the index:

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

If you know the value:

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

The most upvoted answer for `delete` works well in case of objects but not for the real arrays. If I use `delete` it removes elements from loops but keeps the element as `empty` and length of array wont change. This may be a problem in some scenarios.

For example, if I do myArray.toString() on myArray after removal via `delete` it creates empty entry i.e. ,,

answered Jul 14 at 10:24
Arvind K.
501 2 7 19

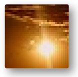
The only working method for me:

```
    while (i < array.length) {
        if(array[i] === value) {
            array.splice(i, 1);
        } else {
            ++i;
        }
    }
    return array;
}
```

usage:

```
var new = removeItem( ["apple","banana", "orange"],  "apple");
// ---> ["banana", "orange"]
```

answered Jul 20 at 20:39



T.Todua
34.9k 12 153 151

Why not use filter instead ? this is a perfect use case for filter – Code Maniac Aug 11 at 16:06