How to get unique values in an array

Asked 8 years, 5 months ago Active 6 months ago Viewed 315k times

179

How can I get a list of unique values in an array? Do I always have to use a second array or is there something similar to java's hashmap in JavaScript?

I am going to be using **JavaScript** and **jQuery** only. No additional libraries can be used.

javascript jquery

39

edited Mar 4 '15 at 21:13

David Sherret

76.5k ● 22 ● 158 ● 157

asked Jun 28 '12 at 14:22

Astronaut

4,959 ● 16 ● 54 ● 89

2 stackoverflow.com/questions/5381621/… - describes exactly what you want I think? – SpaceBison Jun 28 '12 at 14:24

1 are you open to using the `underscore.js` library? – jakee Jun 28 '12 at 14:29

a java hashmap is basically the same as a javascript object. syntax is {"key": "value", "key2": "value2"} – Ian Jun 28 '12 at 14:29

JavaScript/TypeScript collection APIs are terrible compared to Scala, `list.toSet` – Jordan Stewart Dec 12 '19 at 2:36

It depends on what's in the array, how you define "uniqueness" and how large your data is. If they're objects, the code is different than numbers or strings, and if the data is huge, you'll want a linear solution rather than a quadratic one. Please provide more details. – ggorlen Sep 15 at 15:22

20 Answers

Active | Oldest | Votes

120

Since I went on about it in the comments for @Rocket's answer, I may as well provide an example that uses no libraries. This requires two new prototype functions, `contains` and `unique`

```
Array.prototype.contains = function(v) {
  for (var i = 0; i < this.length; i++) {
    if (this[i] === v) return true;
  }
  return false;
};

Array.prototype.unique = function() {
  var arr = [];
  for (var i = 0; i < this.length; i++) {
    if (!arr.contains(this[i])) {
      arr.push(this[i]);
    }
  }
  return arr;
}

var duplicates = [1, 3, 4, 2, 1, 2, 3, 8];
var uniques = duplicates.unique(); // result = [1,3,4,2,8]

console.log(uniques);
```

Run code snippet Expand snippet

For more reliability, you can replace `contains` with MDN's `indexOf` shim and check if each element's `indexOf` is equal to -1: documentation

edited Jul 18 '19 at 16:09
dota2pro
4,628 ● 3 ● 20 ● 47

answered Jun 28 '12 at 14:58

jackwanders
13.8k ● 3 ● 37 ● 39

Thanks for the examples. Ill be using them to filter options for a select box. This should work well. – Astronaut Jun 28 '12 at 16:10

7 this has a high run time complexity (worst case: O(n^2)) – Rahul Arora Oct 21 '17 at 8:40

this is faster than the indexOf approach; no need for the contains polyfill just use includes jsperf.com/get-unique-elements – John Vandivier Nov 6 '17 at 15:12

2 this is a really inefficient implementation. checking the result array to see if it already contains an item is horrible. a better approach would be to either use an object that tracks the counts, or if you dont want to use aux storage, sort it first in O(n log n) then to a linear sweep and compare side by side elements – Isaiah Lee Dec 17 '17 at 2:50

2 Do we really need the "contains" function? – Animesh Kumar Mar 18 '18 at 8:42

218

Or for those looking for a one-liner (simple and functional), **compatible with current browsers**:

```
let a = ["1", "1", "2", "3", "3", "1"];
let unique = a.filter((item, i, ar) => ar.indexOf(item) === i);
console.log(unique);
```

Run code snippet Expand snippet

Update 18-04-2017

It appears as though 'Array.prototype.includes' now has widespread support in the latest versions of the mainline browsers (compatibility)

Update 29-07-2015:

There are plans in the works for browsers to support a standardized 'Array.prototype.includes' method, which although does not directly answer this question; is often related.

Usage:

```
["1", "1", "2", "3", "3", "1"].includes("2");     // true
```

Pollyfill (browser support, source from mozilla):

```
// https://tc39.github.io/ecma262/#sec-array.prototype.includes
if (!Array.prototype.includes) {
```

```
  Object.defineProperty(Array.prototype, 'includes', {
    value: function(searchElement, fromIndex) {

      // 1. Let O be ? ToObject(this value).
      if (this == null) {
        throw new TypeError('"this" is null or not defined');
      }

      var o = Object(this);

      // 2. Let len be ? ToLength(? Get(O, "length")).
      var len = o.length >>> 0;

      // 3. If len is 0, return false.
      if (len === 0) {
        return false;
      }

      // 4. Let n be ? ToInteger(fromIndex).
      //    (If fromIndex is undefined, this step produces the value 0.)
      var n = fromIndex | 0;

      // 5. If n ≥ 0, then
      //  a. Let k be n.
      // 6. Else n < 0,
      //  a. Let k be len + n.
      //  b. If k < 0, let k be 0.
      var k = Math.max(n >= 0 ? n : len - Math.abs(n), 0);

      // 7. Repeat, while k < len
      while (k < len) {
        // a. Let elementK be the result of ? Get(O, ! ToString(k)).
        // b. If SameValueZero(searchElement, elementK) is true, return true.
        // c. Increase k by 1.
        // NOTE: === provides the correct "SameValueZero" comparison needed here.
        if (o[k] === searchElement) {
          return true;
        }
        k++;
      }

      // 8. Return false
      return false;
    }
  });
}
```

edited Oct 30 '19 at 0:26

answered Apr 25 '14 at 0:13

Josh Mc

8,172 ● 6 ● 43 ● 60

It's almost copy paste from kennebec, but admittedly passing the array as a parameter rather than using the closure will probably improve performance. – user1115652 Jan 4 '15 at 21:10

- must say I did not connect the dots, simply scanned through for a one liner, looked like a large post so skipped, went and found an alternative source and re-posted for others to find quickly. That said, your right; pretty much the same as kennebec. – Josh Mc Jan 4 '15 at 22:34

Nice -- Didn't realize that filter sent in the array as a parameter, and didn't want to work on an external object. This is exactly what I need -- my version of javascript (older xerces version) won't have the new goodies for a while. – Gerard ONeill Apr 25 '17 at 23:19

1 @GerardONeill yeah, some situations it is very vital, eg if it is functionally chained and you want access to an array that has not been assigned a variable such as .map(...).filter(...) – Josh Mc Apr 26 '17 at 5:03

1 Terrible answer. O(N^2) complexity. Do not use this. – Timmmm May 14 at 11:35

171

Here's a much cleaner solution for ES6 that I see isn't included here. It uses the Set and the spread operator: `...`

```
var a = [1, 1, 2];

[... new Set(a)]
```

Which returns `[1, 2]`

edited Feb 13 '17 at 7:35

answered Feb 8 '17 at 21:30

Charles Clayton

12.3k ● 10 ● 67 ● 111

1 Thats so clever! – Josh Mc Apr 6 '17 at 22:40

4 Now, **THIS** is a one-liner! – Mac Dec 8 '17 at 20:21

13 In Typescript you have to use `Array.from(new Set(a))` since Set can't be implicitly converted to an array type. Just a heads up! – Zachscs Apr 27 '18 at 21:25

85

One Liner, Pure JavaScript

With ES6 syntax

`list = list.filter((x, i, a) => a.indexOf(x) === i)`

```
x --> item in array
i --> index of item
a --> array reference, (in this case "list")
```

```
> list
<- [1, 3, 4, 1, 2, 1, 3, 3, 4, 1]
> list.filter((x, i, a) => a.indexOf(x) == i);
<- [1, 3, 4, 2]
```

With ES5 syntax

```
list = list.filter(function (x, i, a) {
    return a.indexOf(x) === i;
});
```

Browser Compatibility: IE9+

edited Jun 20 at 9:12

Community ♦

answered Sep 1 '16 at 13:23

Vamsi


1 ● 1


8,342 ● 3 ● 33 ● 43

4 not sure why this was voted down. It may be little obscure at first, and *perhaps* classed as 'clever' and not pragmatic to read, but it's declarative, non-destructive, and concise, where most of the other answers are lacking. – Larry Sep 19 '16 at 15:08

2 @Larry this was down-voted because exactly the same answer was provided years prior to this one. – Alex Okrushko Sep 19 '16 at 21:51

@AlexOkrushko fair enough - missed that answer because of the way it was formatted – Larry Sep 20 '16 at 9:29

10 Everything's a one-liner if you put everything on one line :-) – Gary McGill Sep 18 '17 at 15:42

It might be nice to tighten up the equality `a.indexOf(x) === i` note the three equal signs. – treejanitor Aug 24 '18 at 14:26

22

Using EcmaScript 2016 you can simply do it like this.

```
var arr = ["a", "a", "b"];
var uniqueArray = Array.from(new Set(arr)); // Unique Array ['a', 'b'];
```

Sets are always unique, and using `Array.from()` you can convert a Set to an array. For reference have a look at the documentations.

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

answered Dec 25 '17 at 19:32


Adeel Imran
7,482 ● 4 ● 39 ● 69

1 The is the answer you should use. `indexOf()` answers are terrible because they are O(N^2). The spread answers are ok but won't work for large arrays. This is the best approach. – Timmmm May 14 at 11:37

@Timmmm: Which spread answers won't work for large arrays? – Ry- ♦ Aug 12 at 0:17

Actually I think I was wrong. I was thinking of when people do something like `Math.max(...foo)` but in an array it's ok. `indexOf` is still a terrible idea though! – Timmmm Aug 12 at 13:11

16

These days, you can use ES6's Set data type to convert your array to a unique Set. Then, if you need to use array methods, you can turn it back into an Array:

```
var arr = ["a", "a", "b"];
var uniqueSet = new Set(arr); // {"a", "b"}
var uniqueArr = Array.from(uniqueSet); // ["a", "b"]
//Then continue to use array methods:
uniqueArr.join(", "); // "a, b"
```

answered Mar 7 '16 at 19:58


Fawntasia
331 ● 2 ● 6

1 Neat, would be nice to see some perfomance numbers, I think converting thosesets in particular if they are very dynamic and large could be a performance hickup, only way to tell is to test :) – Astronaut Mar 8 '16 at 1:48

2 If you're using a transpiler or are in an environment that supports it, you can do the same thing more concisely as: `var uniqueArr = [...new Set(arr)]; // ["a", "b"]` – Stenerson Jul 14 '16 at 19:55

Hey @Astronaut have made some tests about performance like you said? – alexventuraio Jul 18 '16 at 21:05

15

If you want to leave the original array intact,

you need a second array to contain the uniqe elements of the first-

Most browsers have `Array.prototype.filter`:

```
var unique= array1.filter(function(itm, i){
    return array1.indexOf(itm)== i;
    // returns true for only the first instance of itm
});


//if you need a 'shim':
Array.prototype.filter= Array.prototype.filter || function(fun, scope){
    var T= this, A= [], i= 0, itm, L= T.length;
    if(typeof fun== 'function'){
        while(i<L){
            if(i in T){
                itm= T[i];
                if(fun.call(scope, itm, i, T)) A[A.length]= itm;
            }
            ++i;
        }
    }
    return A;
}
 Array.prototype.indexOf= Array.prototype.indexOf || function(what, i){
        if(!i || typeof i!= 'number') i= 0;
        var L= this.length;
        while(i<L){
            if(this[i]=== what) return i;
            ++i;
        }
        return -1;
    }
```

edited Jun 28 '12 at 18:55

Rocket Hazmat

198k ● 39 ● 276 ● 318

answered Jun 28 '12 at 14:48

kennebec

91k ● 28 ● 96 ● 123

8

Not native in Javascript, but plenty of libraries have this method.

Underscore.js's `_.uniq(array)` (link) works quite well (source).

answered Jun 28 '12 at 14:29

Calvin

8,057 ● 7 ● 40 ● 49

Thanks for share! This function take iterator and context along with array as an argument list from v1.4.3. – Kunj Mar 15

'17 at 9:13

6

Fast, compact, no nested loops, works with any object not just strings and numbers, takes a predicate, and only 5 lines of code!!

```
function findUnique(arr, predicate) {
  var found = {};
  arr.forEach(d => {
    found[predicate(d)] = d;
  });
  return Object.keys(found).map(key => found[key]);
}
```

Example: To find unique items by type:

```
var things = [
  { name: 'charm', type: 'quark'},
  { name: 'strange', type: 'quark'},
  { name: 'proton', type: 'boson'},
];

var result = findUnique(things, d => d.type);
//  [
//    { name: 'charm', type: 'quark'},
//    { name: 'proton', type: 'boson'}
//  ]
```

If you want it to find the first unique item instead of the last add a found.hasOwnPropery() check in there.

edited Mar 5 '18 at 22:55

answered Dec 4 '17 at 22:18

user1618323

381 ● 5 ● 8

5

Short and sweet solution using second array;

```
var axes2=[1,4,5,2,3,1,2,3,4,5,1,3,4];

    var distinct_axes2=[];

    for(var i=0;i<axes2.length;i++)
        {
        var str=axes2[i];
        if(distinct_axes2.indexOf(str)==-1)
            {
            distinct_axes2.push(str);
            }
        }
    console.log("distinct_axes2 : "+distinct_axes2); // distinct_axes2 : 1,4,5,2,3
```

answered Apr 16 '15 at 4:28

Pradip Shenolkar

769 ● 1 ● 10 ● 29

Short? and sweet? Have you looked at the top solutions? – Alex Okrushko Sep 16 '16 at 20:04

Using jQuery, here's an Array unique function I made:

```
Array.prototype.unique = function () {
    var arr = this;
    return $.grep(arr, function (v, i) {
        return $.inArray(v, arr) === i;
    });
}

console.log([1,2,3,1,2,3].unique()); // [1,2,3]
```

answered Jun 28 '12 at 14:33

Rocket Hazmat

198k • 39 • 276 • 318

5 if you're going to use jQuery inside a core javascript object's prototype, might it not be better to write a jQuery function, such as `$.uniqueArray(arr)`? Embedding references to jQuery within `Array`'s prototype seems questionable – jackwanders Jun 28 '12 at 14:38

1 @jackwanders: What's so questionable about it? If you got jQuery on the page, let's use it. – Rocket Hazmat Jun 28 '12 at 14:40

Just that the new unique function you wrote is now dependent on jQuery; you can't move it to a new site or app without ensuring that jQuery is in use there. – jackwanders Jun 28 '12 at 14:45

2 that was my point; if you're going to use jQuery, then make the function itself part of jQuery. If I were going to extend the prototype of a core object, I'd stick to core javascript, just to keep things reusable. If someone else is looking at your code, it's obvious that `$.uniqueArray` is reliant on jQuery; less obvious that `Array.prototype.unique` is as well. – jackwanders Jun 28 '12 at 14:47

1 @jackwanders: I guess. I use this in my code, as I always use jQuery, and I just like extending `prototype`s. But, I understand your point now. I'll leave this here anyway. – Rocket Hazmat Jun 28 '12 at 14:48

Majority of the solutions above have a high run time complexity.

Here is the solution that uses `reduce` and can do the job in **O(n) time.**

```
Array.prototype.unique = Array.prototype.unique || function() {
        var arr = [];
        this.reduce(function (hash, num) {
                if(typeof hash[num] === 'undefined') {
                        hash[num] = 1;
                        arr.push(num);
                }
                return hash;
        }, {});
        return arr;
}

var myArr = [3,1,2,3,3,3];
console.log(myArr.unique()); //[3,1,2];
```

Run code snippet Expand snippet

Note:

This solution is not dependent on reduce. The idea is to create an object map and push unique ones into the array.

edited Oct 21 '17 at 8:50

answered Oct 21 '17 at 8:45

Rahul Arora

4,157 ● 1 ● 13 ● 22

3

You only need vanilla JS to find uniques with Array.some and Array.reduce. With ES2015 syntax it's only 62 characters.

```
a.reduce((c, v) => b.some(w => w === v) ? c : c.concat(v)), b)
```

Array.some and Array.reduce are supported in IE9+ and other browsers. Just change the fat arrow functions for regular functions to support in browsers that don't support ES2015 syntax.

```
var a = [1,2,3];
var b = [4,5,6];
// .reduce can return a subset or superset
var uniques = a.reduce(function(c, v){
    // .some stops on the first time the function returns true
    return (b.some(function(w){ return w === v; }) ?
      // if there's a match, return the array "c"
      c :
      // if there's no match, then add to the end and return the entire array
      c.concat(v)}),
  // the second param in .reduce is the starting variable. This is will be "c" the
first time it runs.
  b);
```

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

answered Dec 6 '15 at 0:36

puppybits

1,090 ● 11 ● 16

0

You can enter array with duplicates and below method will return array with unique elements.

```
function getUniqueArray(array){
    var uniqueArray = [];
    if (array.length > 0) {
       uniqueArray[0] = array[0];
    }
    for(var i = 0; i < array.length; i++){
        var isExist = false;
        for(var j = 0; j < uniqueArray.length; j++){
            if(array[i] == uniqueArray[j]){
                isExist = true;
                break;
            }
            else{
```

```
                isExist = false;
            }
        }
        if(isExist == false){
            uniqueArray[uniqueArray.length] = array[i];
        }
    }
    return uniqueArray;
}
```

edited Nov 20 '19 at 14:20

answered Oct 23 '18 at 11:17

Muhammad Haroon Iqbal

121 ● 1 ● 4 ● 12

Another thought of this question. Here is what I did to achieve this with fewer code.

0

```
var distinctMap = {};
var testArray = ['John', 'John', 'Jason', 'Jason'];
for (var i = 0; i < testArray.length; i++) {
  var value = testArray[i];
  distinctMap[value] = '';
};
var unique_values = Object.keys(distinctMap);

console.log(unique_values);
```

Run code snippet Expand snippet

edited Jul 18 '19 at 15:20

dota2pro

4,628 ● 3 ● 20 ● 47

answered Jan 4 '16 at 8:26

Xing-Wei Lin

219 ● 3 ● 5

I have tried this problem in pure JS. I have followed following steps 1. Sort the given array, 2. loop through the sorted array, 3. Verify previous value and next value with current value

0

```
// JS
var inpArr = [1, 5, 5, 4, 3, 3, 2, 2, 2,2, 100, 100, -1];

//sort the given array
inpArr.sort(function(a, b){
    return a-b;
});

var finalArr = [];
//loop through the inpArr
for(var i=0; i<inpArr.length; i++){
    //check previous and next value
  if(inpArr[i-1]!=inpArr[i] && inpArr[i] != inpArr[i+1]){
        finalArr.push(inpArr[i]);
  }
}
console.log(finalArr);
```

[Demo]

answered Apr 1 '17 at 11:49

Sanketh Nayak

1

0

```
Array.prototype.unique = function () {
    var dictionary = {};
    var uniqueValues = [];
    for (var i = 0; i < this.length; i++) {
        if (dictionary[this[i]] == undefined){
            dictionary[this[i]] = i;
            uniqueValues.push(this[i]);
        }
    }
    return uniqueValues;
}
```

edited Jan 22 '16 at 15:11

StephenTG

2,411 • 3 • 22 • 35

answered Jan 22 '16 at 14:33

Hexer338

21 • 2

-1

Here is an approach with customizable `equals` function which can be used for primitives as well as for custom objects:

```
Array.prototype.pushUnique = function(element, equalsPredicate = (l, r) => l == r) {
    let res = !this.find(item => equalsPredicate(item, element))
    if(res){
        this.push(element)
    }
    return res
}
```

usage:

```
//with custom equals for objects
myArrayWithObjects.pushUnique(myObject, (left, right) => left.id == right.id)

//with default equals for primitives
myArrayWithPrimitives.pushUnique(somePrimitive)
```

answered Oct 29 '19 at 14:03

vir us

6,557 • 2 • 38 • 52

-1

If you don't need to worry so much about older browsers, this is exactly what Sets are designed for.

> The Set object lets you store unique values of any type, whether primitive values or object references.

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

```
const set1 = new Set([1, 2, 3, 4, 5, 1]);
// returns Set(5) {1, 2, 3, 4, 5}
```

answered Nov 9 '18 at 11:43

Paul Cooper

277 ● 2 ● 12

-2

I was just thinking if we can use linear search to eliminate the duplicates:

```
JavaScript:
function getUniqueRadios() {

var x=document.getElementById("QnA");
var ansArray = new Array();
var prev;


for (var i=0;i<x.length;i++)
  {
    // Check for unique radio button group
    if (x.elements[i].type == "radio")
    {
            // For the first element prev will be null, hence push it into array and
set the prev var.
            if (prev == null)
            {
                prev = x.elements[i].name;
                ansArray.push(x.elements[i].name);
            } else {
                   // We will only push the next radio element if its not identical to
previous.
                   if (prev != x.elements[i].name)
                   {
                       prev = x.elements[i].name;
                       ansArray.push(x.elements[i].name);
                   }
            }
    }

  }

   alert(ansArray);
```

}

HTML:

```
<body>

<form name="QnA" action="" method='post' ">

<input type="radio"  name="g1" value="ANSTYPE1"> good </input>
<input type="radio" name="g1" value="ANSTYPE2"> avg </input>

<input type="radio"  name="g2" value="ANSTYPE3"> Type1 </input>
<input type="radio" name="g2" value="ANSTYPE2"> Type2 </input>
```

```
<input type="submit" value='SUBMIT' onClick="javascript:getUniqueRadios()"></input>


</form>
</body>
```

answered Aug 12 '13 at 10:28

Suresh

1,367 ● 1 ● 8 ● 4

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.