How to get a key in a JavaScript object by its value?

Asked 7 years, 6 months ago Active 18 days ago Viewed 413k times

310

I have a quite simple JavaScript object, which I use as an associative array. Is there a simple function allowing me to get the key for a value, or do I have to iterate the object and find it out manually?

javascript object

86

edited Oct 24 '17 at 10:28

Sebastian Simon
11.3k 6 33 54

asked Mar 28 '12 at 12:23

arik
11.6k 29 85 138

There is no such standard function to do this. If the mapping is truly bidirectional then it is trivial to construct a "flipped" map and index that. Otherwise a simple property-iterator (with a hasOwnProperty gaurd, perhaps) and an early-return hidden inside a function does just nicely... – user166390 Mar 28 '12 at 12:27

How could this work if an object was referenced by more than one key? `var o = []; var map = {first: o, second: o}` . What would `find_key(o)` return? – Gareth Mar 28 '12 at 12:49

2 doesn't matter ;) I only intended to use it for an array with unique key-value-pairs. – arik Mar 30 '12 at 20:31

Possible duplicate of best way to get the key of a key/value javascript object – Andy Feb 4 '16 at 18:02

I've made a version without iteration stackoverflow.com/a/36705765/696535. It would be interesting to test all proposed solutions in jsfiddle – Pawel Feb 1 '18 at 22:04

25 Answers

with `Underscore.is` librarv:

```
  foo: 1,
  bar: 2
};
```

```
(_.invert(hash))[1]; // => 'foo'
```

edited Jul 19 '18 at 16:18 — T.Todua — 34.7k 12 153 151

answered Aug 26 '13 at 2:36 — banyan — 1,907 2 17 15

259	@GeorgeJempty Not everyone wants to load a 5kb library for a simple key lookup ;) – Doorknob Jan 12 '14 at 2:45
3	Just FYI for anyone looking for a solution that will get you ALL keys that match a value: this will not work. – Brett Mar 20 '14 at 6:45
2	underscore keys will work too. underscorejs.org/#keys _.keys({one: 1, two: 2, three: 3}); => ["one", "two", "three"] – Thaddeus Albers May 15 '14 at 19:54
1	Is there an equivalent in angularjs? – Inigo Jul 11 '14 at 21:31
65	"You can do ___ with ___ library" is never a good answer. – Terra Ashley Jul 29 '16 at 3:48

441

```
function getKeyByValue(object, value) {
  return Object.keys(object).find(key => object[key] === value);
}
```

ES6, no prototype mutations or external libraries.

Example,

```
function getKeyByValue(object, value) {
  return Object.keys(object).find(key => object[key] === value);
}


const map = {"first" : "1", "second" : "2"};
console.log(getKeyByValue(map,"2"));
```

edited Mar 22 at 13:03

jithinkmatthew
454 5 13

answered Jan 28 '15 at 12:11

UncleLaz
5,028 2 11 11

5 Brilliant! Quite the cleanest way - if your browser supports it (mine does :) - Chrome 54) – Velojet Oct 28 '16 at 4:14

6 Well, really clean if you don't support IE11 :-)If so, you need a polyfill – Chexpir Mar 2 '17 at 10:23

4 Depending on the implementation, this probably takes *O(n)* space since `keys()` materializes the key set. – David Ehrmann Oct 19 '17 at 1:04

2 Clean but slow. – Dirigible Dec 10 '17 at 17:35

2 If Multiple keys have Same value use filter instead of find `function getKeyByValue(object, value) { return Object.keys(object).filter(key => object[key] === value); }` – saketh May 28 at 14:22

172

No standard method available. You need to iterate and you can create a simple helper:

```
Object.prototype.getKeyByValue = function( value ) {
    for( var prop in this ) {
        if( this.hasOwnProperty( prop ) ) {
             if( this[ prop ] === value )
                 return prop;
        }
    }
}

var test = {
   key1: 42,
   key2: 'foo'
};

test.getKeyByValue( 42 );  // returns 'key1'
```

One word of caution: Even if the above works, its generally a bad idea to extend any host or native object's `.prototype`. I did it here because it fits the issue very well. Anyway, you should probably use this function outside the `.prototype` and pass the object into it instead.

 **185k** 44 276 335

Ok, thanks! I hope they'll create a "native" function for that one day, though. – arik Mar 28 '12 at 19:50

2 Actually it's ok if you know things like that the for-in loop goes down the property chain which means "for(var key in obj)" would give you "getKeyByValue" as "key" at some point. – user659025 Oct 22 '12 at 15:16

I noticed that this method attaches to each and every object in the source. Although there's a word of caution, I don't think many realize that it is copied a zillion times. – wubbewubbewubbe Mar 7 '13 at 11:32

2 @wubbewubbewubbe: its not copied, that is the beauty of prototypal inheritance. Think more like 'delegation' of it. However, I guess I did the warning very clearly, not because there is a problem of memory or performance, but you never know if you overwrite or shadow someone elses code. – jAndy Mar 7 '13 at 11:34

1 @jAndy it is NOT ===, it is ==. Your code does not work with ===. It returns undefined. – Dexter Jan 6 '15 at 19:44

102

As said, iteration is needed. For instance, in modern browser you could have:

```
var key = Object.keys(obj).filter(function(key) {return obj[key] === value})[0];
```

Where `value` contains the value you're looking for. Said that, I would probably use a loop.

Otherwise you could use a proper "hashmap" object - there are several implementation in JS around - or implement by your own.

UPDATE 2018

Six years passed, but I still get some vote here, so I feel like a more modern solution – for modern browser/environment – should be mentioned in the answer itself and not just in the comments:

```
const key = Object.keys(obj).find(key => obj[key] === value);
```

Of course it can be also a function:

```
const getKeyByValue = (obj, value) =>
        Object.keys(obj).find(key => obj[key] === value);
```


18.9k 4 38 45

15 ES6: `Object.keys(obj).or(o=>o[key] === value)` – Benjamin Gruenbaum Jun 23 '13 at 11:54

Unfortunately the arrow function is not any "modern" browser yet, so it's a bit useless at the moment – I'm using it in jetpack on Firefox Nightly, it will be in Firefox 22. Anyway, I'm not aware about any `or` array's method, and it's not clear to me its purpose here: I will appreciate some additional detail! :) – ZER0 Jun 23 '13 at 18:40

1 As for arrow, it's coming and I'm waiting for it :) As for `or` sure! It was only recently evaluated and accepted (I don't think anyone implements it yet). What it does is find the first element of an array matching a predicate and return it. So `[1,2,3,4].or(x=>x>2)` would return `3` and `[1,2,3,4,5].or(x=>x<3)` would return `1` . Something like C#'s FirstOrDefault :) – Benjamin Gruenbaum Jun 23 '13 at 18:43

Yeah, arrow is coming but it will takes to be used widely – unless as I do, someone's working on a specific engine. I wasn't aware of new proposal for ES6, I thought was pretty closed: do you have a link about the `or` method? From what you mentioned it seems that it returns the item that match the predicate "or" the array itself? – ZER0 Jun 24 '13 at 6:56

Looks like it was renamed to `Array.prototype.find` and `Array.prototype.findIndex` (for finding the index rather than the element :) github.com/Ralt/or/issues/1#issuecomment-14940411 – Benjamin Gruenbaum Jun 24 '13 at 7:16

40

The lodash way https://lodash.com/docs#findKey

```
var users = {
  'barney':  { 'age': 36, 'active': true },
  'fred':    { 'age': 40, 'active': false },
  'pebbles': { 'age': 1,  'active': true }
};

_.findKey(users, { 'age': 1, 'active': true });
// → 'pebbles'
```

Run code snippet Expand snippet

answered Mar 20 '16 at 1:08


cantbebothered
441 4 2

4 FYI, "the underscore way": `_.findKey(users, { 'age': 1, 'active': true });` ...it's the same – craigmichaelmartin Jul 19 '16 at 12:18

7 if your values are simple, like strings or integers, then contrary to expectation this will not work. e.g. `_.find_key({a: "A", b:"B"}, "B"})`

returns `undefined` so as stated here you need to do `_.find_key({a: "A", b:"B"}, _.partial(_.isEqual,"B")})` – ryan2johnson9 Jan 9 '17 at 4:17

1 When you don't have lodash stackoverflow.com/a/36705765/696535 – Pawel Jan 16 '17 at 10:55

1 @ryan2johnson9 That's my problem with Lodash. I'm having a hard time understanding some functions (apparently I'm the only one). But thanks anyway, it works. I found another, shorter solution. It causes overheap on bigger objects so be careful with this one. `_.invert(haystack)[needle]` – Empi Apr 3 '17 at 15:33

31

```
function extractKeyValue(obj, value) {
    return Object.keys(obj)[Object.values(obj).indexOf(value)];
}
```

Made for closure compiler to extract key name which will be unknown after compilation

More sexy version but using future `Object.entries` function

```
function objectKeyByValue (obj, val) {
  return Object.entries(obj).find(i => i[1] === val);
}
```

edited Jan 16 '17 at 11:12

answered Apr 18 '16 at 23:02

Pawel
6,568 3 41 48

6 I think this is the best one for 2017+ since it uses plain JavaScript. – brainbag Feb 12 '18 at 15:46

Doesn't seem to work if you have two or more numbers that have the same value – JuicY_Burrito Jul 4 at 11:30

@SamuelChen that's right but if it worked it would mean an array is needed as a result. Where `Object.entries(obj).find(i => i[1] === val);` use `filter` instead `Object.entries(obj).filter(i => i[1] === val);` – Pawel Jul 6 at 9:29

21

```
Object.prototype.getKey = function(value){
  for(var key in this){
    if(this[key] == value){
      return key;
    }
  }
  return null;
};
```

Usage:

```
// ISO 639: 2-letter codes
var languageCodes = {
  DA: 'Danish',
  DE: 'German',
  DZ: 'Bhutani',
  EL: 'Greek',
  EN: 'English',
  EO: 'Esperanto',
  ES: 'Spanish'
};

var key = languageCodes.getKey('Greek');
console.log(key); // EL
```

answered Aug 3 '12 at 13:32

Benny Neugebauer
30.7k 17 162 159

10 +1 neat solution. But i have a question: Shouldn't you always check for `obj.hasOwnProperty(key)` or is it unnecessary in this case ? – V-Light Jun 12 '13 at 9:38

3 Mutating the Object prototype is bad practice: stackoverflow.com/questions/23807805/… – Jon Koops Jan 6 '16 at 17:49

16

> Keep your ***prototype*** clean.

```
                return key;
            }
        }

     return false;
    }
```

Example:

```
var map = {"first" : 1, "second" : 2};
var key = val2key(2,map); /*returns "second"*/
```

edited Dec 12 '17 at 0:04

answered Dec 13 '15 at 17:01

Kareem

2,916 29 29

14

Non-iteratable solution

Main function:

```
var keyByValue = function(value) {

    var kArray = Object.keys(greetings);        // Creating array of keys
    var vArray = Object.values(greetings);      // Creating array of values
    var vIndex = vArray.indexOf(value);         // Finding value index

    return kArray[vIndex];                      // Returning key by value index
}
```

Object with keys and values:

```
var greetings = {
    english   : "hello",
    ukranian  : "привіт"
};
```

```
keyByValue("привіт");
// => "ukranian"
```

edited Mar 13 '17 at 12:54

answered Mar 13 '17 at 12:45

Systems Rebooter

408 5 15

14

If you are working with **Underscore** or **Lodash** library, you can use the _.findKey function:

```
var users = {
  'barney':  { 'age': 36, 'active': true },
  'fred':    { 'age': 40, 'active': false },
  'pebbles': { 'age': 1,  'active': true }
};

_.findKey(users, function(o) { return o.age < 40; });
// => 'barney' (iteration order is not guaranteed)

// The `_.matches` iteratee shorthand.
_.findKey(users, { 'age': 1, 'active': true });
// => 'pebbles'

// The `_.matchesProperty` iteratee shorthand.
_.findKey(users, ['active', false]);
// => 'fred'

// The `_.property` iteratee shorthand.
_.findKey(users, 'active');
// => 'barney'
```

edited May 5 at 9:00

Gil Epshtain

2,552 4 26 49

answered Aug 4 '15 at 17:59

craigmichaelmartin

2,398 1 13 16

I created the *bimap* library (https://github.com/alethes/bimap) which implements a powerful, flexible and efficient JavaScript bidirectional map interface. It has no dependencies and is usable both on the server-side (in Node.js, you can install it with `npm install bimap`) and in

```
var bimap = new BiMap;
bimap.push("k", "v");
bimap.key("k") // => "v"
bimap.val("v") // => "k"

bimap.push("UK", ["London", "Manchester"]);
bimap.key("UK"); // => ["London", "Manchester"]
bimap.val("London"); // => "UK"
bimap.val("Manchester"); // => "UK"
```

Retrieval of the key-value mapping is equally fast in both directions. There are no costly object/array traversals under the hood so the average access time remains constant regardless of the size of the data.

edited May 19 '14 at 1:40

answered Apr 22 '14 at 2:09

Alethes
792 6 8

3

Since the values are unique, it should be possible to add the values as an additional set of keys. This could be done with the following shortcut.

```
var foo = {};
foo[foo.apple = "an apple"] = "apple";
foo[foo.pear = "a pear"] = "pear";
```

This would permit retrieval either via the key or the value:

```
var key = "apple";
var value = "an apple";

console.log(foo[value]); // "apple"
console.log(foo[key]); // "an apple"
```

This does assume that there are no common elements between the keys and values.

answered Aug 20 '13 at 11:32

Browsing all the solutions, I went with this one. Very intuitive. – nehemiah Feb 19 '18 at 11:22

2

Or, easier yet - create a new object with the keys and values in the order you want then do look up against that object. We have had conflicts using the prototype codes above. You don't have to use the String function around the key, that is optional.

```
newLookUpObj = {};
$.each(oldLookUpObj,function(key,value){
       newLookUpObj[value] = String(key);
   });
```

answered Mar 28 '13 at 17:59

Slopeside Creative
3,528 4 12 17

2

As if this question hasn't been beaten to a pulp...

Here's one just for whatever curiosity it brings you...

If you're sure that your object will have only string values, you could really exhaust yourself to conjure up this implementation:

```
var o = { a: '_A', b: '_B', c: '_C' }
  , json = JSON.stringify(o)
  , split = json.split('')
  , nosj = split.reverse()
  , o2 = nosj.join('');

var reversed = o2.replace(/[{}]+/g, function ($1) { return ({ '{':'}', '}':'{' })[$1];
})
  , object = JSON.parse(reversed)
  , value = '_B'
  , eulav = value.split('').reverse().join('');

console.log('>>', object[eulav]);
```

answered Apr 28 '16 at 5:16

Cody

7,527 3 46 38

1

http://jsfiddle.net/rTazZ/2/

```
var a = new Array();
    a.push({"1": "apple", "2": "banana"});
    a.push({"3": "coconut", "4": "mango"});

    GetIndexByValue(a, "coconut");

    function GetIndexByValue(arrayName, value) {
    var keyName = "";
    var index = -1;
    for (var i = 0; i < arrayName.length; i++) {
       var obj = arrayName[i];
            for (var key in obj) {
                if (obj[key] == value) {
                    keyName = key;
                    index = i;
                }
            }
        }
        //console.log(index);
        return index;
    }
```

answered Jan 26 '13 at 8:07

Atur

984 6 21 34

performance impacts on huge arrays? – Fr0zenFyr Jun 10 '15 at 21:18

@Fr0zenFyr : Following link can answer you question better - stackoverflow.com/questions/8423493/… – Atur Jun 11 '15 at 5:42

1

```
var getKeyByValue = function(searchValue) {
  return _.findKey(hash, function(hashValue) {
    return searchValue === hashValue;
  });
}
```

FindKey will search and return the first key which matches the value.
If you want the last match instead, use FindLastKey instead.

answered Jan 27 '14 at 19:41

kumarharsh
13.9k 6 59 87

I typically recommend lodash rather than underscore.

1

If you have it, use it.

If you don't, then you should consider using the lodash.invert npm package, which is pretty tiny.

Here's how you can test it using gulp:

1) Create a file called gulpfile.js with the following contents:

```
// Filename: gulpfile.js
var gulp = require('gulp');
var invert = require('lodash.invert');
gulp.task('test-invert', function () {
  var hash = {
    foo: 1,
    bar: 2
  };
  var val = 1;
  var key = (invert(hash))[val];  // << Here's where we call invert!
  console.log('key for val(' + val + '):', key);
});
```

2) Install the lodash.invert package and gulp

3) Test that it works:

```
$ gulp test-invert
[17:17:23] Using gulpfile ~/dev/npm/lodash-invert/gulpfile.js
[17:17:23] Starting 'test-invert'...
key for val(1): foo
[17:17:23] Finished 'test-invert' after 511 µs
```

References

https://www.npmjs.com/package/lodash.invert

https://lodash.com/

Differences between lodash and underscore

https://github.com/gulpjs/gulp

edited May 23 '17 at 11:55
Community ♦
1 1

answered Jun 3 '15 at 21:26
l3x
21.8k 1 41 31

1

Given `input={"a":"x", "b":"y", "c":"x"}` ...

To use the first value (e.g. `output={"x":"a","y":"b"}`): `output=Object.keys(input).reduceRight(function(accum,key,i){accum[input[key]]=key;return accum;},{})`

To use the last value (e.g. `output={"x":"c","y":"b"}`): `output=Object.keys(input).reduce(function(accum,key,i){accum[input[key]]=key;return accum;},{})`

To get an array of keys for each value (e.g. `output={"x":["c","a"],"y":["b"]}`):

```
output=Object.keys(input).reduceRight(function(accum,key,i){accum[input[key]]=(accum[input[key]]||[]).concat(key);return accum;},{})
```

answered Jul 2 '15 at 15:02
Fabio Beltramini

this is definitely the best answer , but I was scratching my head over a way to transform it in order to return only the key for a given object, ie be functionnally equivalent to indexOf for an array. – Souhaieb Besbes Apr 13 '16 at 10:28

Unless memory is a constraint and you are willing to spend a lot of processing power to look through the object many times over, just save the "output" as indicated above to a variable and look up the result in there... like `output['x']` . Is that what you were asking? – Fabio Beltramini Apr 16 '16 at 19:17

1

Here's a Lodash solution to this that works for flat key => value object, rather than a nested object. The accepted answer's suggestion to use `_.findKey` works for objects with nested objects, but it doesn't work in this common circumstance.

This approach inverts the object, swapping keys for values, and then finds the key by looking up the value on the new (inverted) object. If the key isn't found then `false` is returned, which I prefer over `undefined` , but you could easily swap this out in the third parameter of the `_.get` method in `getKey()` .

```
// Get an object's key by value
var getKey = function( obj, value ) {
        var inverse = _.invert( obj );
        return _.get( inverse, value, false );
};

// US states used as an example
var states = {
        "AL": "Alabama",
        "AK": "Alaska",
        "AS": "American Samoa",
        "AZ": "Arizona",
        "AR": "Arkansas",
        "CA": "California",
        "CO": "Colorado",
        "CT": "Connecticut",
        "DE": "Delaware",
        "DC": "District Of Columbia",
        "FM": "Federated States Of Micronesia",
        "FL": "Florida",
        "GA": "Georgia",
        "GU": "Guam",
        "HI": "Hawaii",
        "ID": "Idaho",
        "IL": "Illinois",
        "IN": "Indiana",
```

```
        "LA": "Louisiana",
        "ME": "Maine",
        "MH": "Marshall Islands",
        "MD": "Maryland",
        "MA": "Massachusetts",
        "MI": "Michigan",
        "MN": "Minnesota",
        "MS": "Mississippi",
        "MO": "Missouri",
        "MT": "Montana",
        "NE": "Nebraska",
        "NV": "Nevada",
        "NH": "New Hampshire",
        "NJ": "New Jersey",
        "NM": "New Mexico",
        "NY": "New York",
        "NC": "North Carolina",
        "ND": "North Dakota",
        "MP": "Northern Mariana Islands",
        "OH": "Ohio",
        "OK": "Oklahoma",
        "OR": "Oregon",
        "PW": "Palau",
        "PA": "Pennsylvania",
        "PR": "Puerto Rico",
        "RI": "Rhode Island",
        "SC": "South Carolina",
        "SD": "South Dakota",
        "TN": "Tennessee",
        "TX": "Texas",
        "UT": "Utah",
        "VT": "Vermont",
        "VI": "Virgin Islands",
        "VA": "Virginia",
        "WA": "Washington",
        "WV": "West Virginia",
        "WI": "Wisconsin",
        "WY": "Wyoming"
};

console.log( 'The key for "Massachusetts" is "' + getKey( states, 'Massachusetts' ) +
'"' );


<script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.4/lodash.min.js">
</script>
```

answered Sep 28 '17 at 16:40

Kevin Leary
4,723 36 40

1

Here is my solution first:

For example, I suppose that we have an object that contains three value pairs:

```
function findKey(object, value) {

    for (let key in object)
        if (object[key] === value) return key;

    return "key is not found";
}

const object = { id_1: "apple", id_2: "pear", id_3: "peach" };

console.log(findKey(object, "pear"));
//expected output: id_2
```

We can simply write a findKey(array, value) that takes two parameters which are an object and the value of the key you are looking for. As such, this method is reusable and you do not need to manually iterate the object every time by only passing two parameters for this function.

edited Oct 5 '18 at 11:52

answered Oct 5 '18 at 11:34

Ivan Zhang
11 2

0

Underscore js solution

```
let samplLst = [{id:1,title:Lorem},{id:2,title:Ipsum}]
let sampleKey =  .findLastIndex(samplLst,{ id:2});
```

answered Mar 23 '18 at 11:48

Pooja
306 2 9

0

Really straightforward.

```
const CryptoEnum = Object.freeze({
                    "Bitcoin": 0, "Ethereum": 1,
                    "Filecoin": 2, "Monero": 3,
                    "EOS": 4, "Cardano": 5,
                    "NEO": 6, "Dash": 7,
                    "Zcash": 8, "Decred": 9
                   });

Object.entries(CryptoEnum)[0][0]
// output => "Bitcoin"
```

answered May 22 '18 at 12:42

onepix
154 1 9

6 No guarantee that the object order will be the same... – Downgoat Jul 11 '18 at 4:41

0

ES6 methods:

```
Object.fromEntries(Object.entries(a).map(b => b.reverse()))['value_you_look_for']
```

edited Sep 10 at 6:09

answered Sep 10 at 6:03

atl3
62 5

0

```
let key = ( Object.entries(obj).find(([k,v])=>v==value) || [] )[0];
```

answered Sep 15 at 13:56

rturkek

811 10 17

-1

Keep it simple!

You don't need to filter the object through sophisticated methods or libs, Javascript has a built in function called **Object.values**.

Example:

```
let myObj = {jhon: {age: 20, job: 'Developer'}, marie: {age: 20, job:
'Developer'}};

function giveMeTheObjectData(object, property) {
   return Object.values(object[property]);
}

giveMeTheObjectData(myObj, 'marie'); // => returns marie: {}
```

This will return the object property data.

References

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Object/values

edited Nov 7 '18 at 12:03

Ore4444

7,287 2 16 26

answered Sep 10 '18 at 13:17

user6358125

---

protected by adiga Jun 12 at 12:12

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