Reverse lookup object with array

Asked 5 years ago Active 2 years, 7 months ago Viewed 3k times

3

Let say if I have an object like this

```
resourceMap = {
    "a": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    "b": [11, 12],
    "c": [21, 23],
    "d": [54, 55, 56, 57, 510]
};
```

What is the best way to figure out if `resourceId = 21` would be `"c"` ?

We don't know the key names or number of keys. It only matches once: meaning `21` will belong to only one key `"c"` .

I am thinking of looping through all keys and do `indexOf()` , but I don't feel it's "elegant" enough.

I could use Underscore but try to avoid and go with what Angular or jQuery or just vanilla Javascript.

javascript jquery json angularjs

edited Oct 7 '14 at 19:58

asked Oct 7 '14 at 17:10

HP.

7,921 40 125 223

Are the numbers in the arrays distinct, or is it possible that a number reverse-maps to more than a single item? – spender Oct 7 '14 at 17:12

Either you pre-process it to be in a different format or you loop on demand. Either way Vanilla JS or underscore, you will end up doing the same thing under the covers. – epascarello Oct 7 '14 at 17:13

@spender says it right in the OPs question. *"It only matches once:..."* – epascarello Oct 7 '14 at 17:13

5 Answers

7

It's perfectly acceptable to have numeric property names for objects in JavaScript. We can use this to our advantage to build a second object that maps everything in reverse. This will make lookups inexpensive.

```
var reverseMap = {};
for(var propName in resourceMap)
{
    var numsArr = resourceMap[propName];
    numsArr.forEach(function(num){
        reverseMap[num]=propName;
    });
}
console.log(reverseMap[54]); //'d'
```

http://jsfiddle.net/y11sbgbv/

Building the reverseMap can also be done more "functionally" (e.g. without using side-effects) as follows:

```
var reverseMap2 = Object.keys(resourceMap).reduce((acc, propName) =>
  resourceMap[propName].reduce((a, num) => {
    a[num] = propName;
    return a;
  }, acc), {});
```

edited Feb 22 '17 at 21:13

answered Oct 7 '14 at 17:17

spender
91.1k 23 176 296

3

Preprocess into a different look up table

```
var revMap = []; // or var revMap = {};
```

```
});
console.log(revMap[21])
```

or look up on demand:

```
resourceMap = { "a": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], "b": [11, 12], "c": [21, 23],
"d": [54, 55, 56, 57, 510] };
function findMapKey (map, ind) {
    var match = null;
    Object.keys(map).some(function(key) {
        var test = map[key].indexOf(ind)!==-1;
        if (test) {
            match = key;
        }
        return test;
    });
    return match;
}
console.log(findMapKey(resourceMap, 21));
```

answered Oct 7 '14 at 17:32

epascarello

161k 15 146 192

`Object.keys` ... didn't know about that. +1 , although the sparseness of arrays is an implementation detail that I wouldn't rely on. Given that numeric property names are fine, I'd use an object over an array for the map. – spender Oct 7 '14 at 17:33

0

It looks like values are sorted, so you can first look if the latest of each array is greater than the value you're searching for, once you find it, look for the value itself.

```
for(var propName in resourceMap)
{
    var array = resourceMap[propName];
    var lastNum = array[array.length-1];

    if(lastNum == 21 || (lastNum > 21 && array.indexOf(21) > -1))
    {
```

```
}
```

answered Oct 7 '14 at 17:23

Luizgrs
3,430 1 13 21

if you're going to do searches on it often, then go with spender answer – Luizgrs Oct 7 '14 at 17:26

0

Unless you're working with a massive amount of data, your idea is fine:

```
var resourceMap = {
  "a": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
  "b": [11, 12],
  "c": [21, 23],
  "d": [54, 55, 56, 57, 510]
};

/**
 * @return resource's group, or false if no matching group is found
 */
var getResourceGroup = function(res) {
  for (var i in resourceMap) {
    if (resourceMap[i].indexOf(res) > -1) {
      return i;
    }
  }
  return false;
}

console.log(getResourceGroup(21));
```

Run code snippet Expand snippet

I understand the desire for elegance, and I strive for that in my own work as well. But this is simple, easily supported, and relatively fast assuming your data set isn't huge.

large data structure in memory. Probably not a big deal, but worth considering.

edited Oct 7 '14 at 17:44

answered Oct 7 '14 at 17:17

Madbreaks

14.9k 5 43 60

0

Try

```
resourceMap = {
    "a": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    "b": [11, 12],
    "c": [21, 23],
    "d": [54, 55, 56, 57, 510]
};

var getKey = function(val, _key) {
$.each(resourceMap, function(k, v) {
  if ($.inArray(val, v) != -1) {
    _key = k
  }
})
return _key
};

getKey(21)
```

jsfiddle http://jsfiddle.net/guest271314/gbnzmq51/

answered Oct 7 '14 at 17:51

guest271314

1
