Swap key with value JSON

Asked 5 years, 6 months ago Active 4 days ago Viewed 49k times

82

I have an extremely large JSON object structured like this:

```
{A : 1, B : 2, C : 3, D : 4}
```

I need a function that can swap the values with keys in my object and I don't know how to do it. I would need an output like this:

12

```
{1 : A, 2 : B, 3 : C, 4 : D}
```

Is there any way that I can do this would manually created a new object where everything is swapped?
Thanks

javascript json node.js key-value

asked Apr 11 '14 at 13:11

C1D
1,471 1 15 19

1 are all values numbers and do the numbers repeat? If the values repeat then you wont be able to swap them as they will overwrite the others, unless you change their value to some unique value. There might be a better solution what is the reason for needing the swap? – Patrick Evans Apr 11 '14 at 13:13

@PatrickEvans They're all numbers but they don't repeat. I'm trying to make a basic cipher. – C1D Apr 11 '14 at 13:15

1 It's worth noting that a "swap" operation like this is problematic for [at least] two reasons: 1) Values are cast to Strings when they become keys so don't be surprised when you have unexpected "[object Object]" keys. And 2) duplicate values (after cast to String) get overwritten. #1, at least, can be solved by producing a Map instead of an Object, thusly: `function swap(obj) {return new Map(Object.entries(x).map([k, v] => [v, k]))}` – broofa Feb 1 at 21:44

13 Answers

90

```
function swap(json){
  var ret = {};
  for(var key in json){
    ret[json[key]] = key;
  }
  return ret;
}
```

Example here FIDDLE don't forget to turn on your console to see the results.

ES6 versions:

```
static objectFlip(obj) {
  const ret = {};
  Object.keys(obj).forEach(key => {
    ret[obj[key]] = key;
  });
  return ret;
}
```

Or using Array.reduce() & Object.keys()

```
static objectFlip(obj) {
  return Object.keys(obj).reduce((ret, key) => {
    ret[obj[key]] = key;
    return ret;
  }, {});
}
```

Or using Array.reduce() & Object.entries()

```
static objectFlip(obj) {
  return Object.entries(obj).reduce((ret, entry) => {
    const [ key, value ] = entry;
    ret[ value ] = key;
    return ret;
  }, {});
}
```


KostasX
639 6 15


jPO
1,406 11 20

34

you can use [lodash function _.invert](#) it also can use multivlaue

```
var object = { 'a': 1, 'b': 2, 'c': 1 };

_.invert(object);
// => { '1': 'c', '2': 'b' }

// with `multiValue`
_.invert(object, true);
// => { '1': ['a', 'c'], '2': ['b'] }
```

edited Mar 14 '18 at 14:05


Edo
2,064 1 18 19

answered Jun 8 '16 at 8:41


Ohad Sadan
524 4 10

33

Get the keys of the object, and then use the Array's reduce function to go through each key and set the value as the key, and the key as the value.

```
var data = {A : 1, B : 2, C : 3, D : 4}
var newData = Object.keys(data).reduce(function(obj,key){
   obj[ data[key] ] = key;
   return obj;
},{});
console.log(newData);
```

answered Apr 11 '14 at 13:17


Patrick Evans
33.9k 6 55 76

29

```
const obj = { a: "aaa", b: "bbb", c: "ccc", d: "ddd" };
Object.assign({}, ...Object.entries(obj).map(([a,b]) => ({ [b]: a })))
```

answered Oct 5 '17 at 9:54

grappeq

315 3 6

1 Seems **the better solution** for nowdays (2019)! Somebody can confirm, is it? Performance good? Semantic is perfect: we can see that is really a simple "map and swap" solution. – Peter Krauss Jan 3 at 17:41

@peter-krauss, You are welcome to tinker with jsben.ch/gHEtn in case I missed something, but a casual comparison between this answer and jPO's answer shows that jPO's for-loop outperforms grappeq's map approach by almost 3 to 1 (at least on *my* browser). – Michael Hays Feb 20 at 17:45

28

In ES6/ES2015 you can combine use of Object.keys and reduce with the new Object.assign function, an arrow function, and a computed property name for a pretty straightforward single statement solution.

```
const foo = { a: 1, b: 2, c: 3 };
const bar = Object.keys(foo)
    .reduce((obj, key) => Object.assign({}, obj, { [foo[key]]: key }), {});
```

If you're transpiling using the object spread operator (stage 3 as of writing this) that will simplify things a bit further.

```
const foo = { a: 1, b: 2, c: 3 };
const bar = Object.keys(foo)
    .reduce((obj, key) => ({ ...obj, [foo[key]]: key }), {});
```

Finally, if you have Object.entries available (stage 4 as of writing), you can clean up the logic a touch more (IMO).

```
const foo = { a: 1, b: 2, c: 3 };
const bar = Object.entries(foo)
    .reduce((obj, [key, value]) => ({ ...obj, [value]: key }), {});
```

answered May 10 '17 at 17:58

SyntaxError: /Users/markus/Entwicklung/IT1_Beleg/public/es6/vokabeltrainer.js: Unexpected token (53:45) 51 | if (btoa) { 52 | entries = Object.entries(entries) > 53 | .reduce((obj, [key, value]) => ({...obj, [value]: key}), {}); | ^ – Superlokkus Jun 16 '17 at 19:32

@Superlokkus my best interpretation of that error output is that you're not transpiling the object spread syntax, and as of today, I don't know of any JavaScript engines that support it natively. – joslarson Jul 26 '17 at 14:47

1 See @grappeq's solution, seems better (the simplest!). – Peter Krauss Jan 3 at 17:31

8

Now that we have Object.fromEntries:

```
obj => Object.fromEntries(Object.entries(obj).map(a => a.reverse()))
```

or

```
obj => Object.fromEntries(Object.entries(obj).map(([k, v]) => ([v, k])))
```

answered Jun 26 at 21:37

Sc0ttyD
662 1 8 20

Should be standard way of doing object swap as of Node.js version 12 as well. – Damaged Organic Jul 16 at 7:05

4

As a complement of @joslarson and @jPO answers:
Without ES6 needed, you can use `Object.keys` `Array.reduce` and the Comma Operator:

```
Object.keys(foo).reduce((obj, key) => (obj[foo[key]] = key, obj), {});
```

Some may find it ugly, but it's "kinda" quicker as the `reduce` doesn't spread all the properties of the `obj` on each loop.

answered Dec 27 '17 at 16:48

Vaidd4

Still... jPO's answer outperforms this one by almost 2:1. jsben.ch/R75aI (I know your comment was from long, long ago, but I was commenting on grappeq's answer and figured I'd run your example as well). – Michael Hays Feb 20 at 17:51

Totally, a for loop remains faster then the `forEach`, `map` or `reduce` methods. I made this response to have a one-liner solution without the need of es6 and still be relevant in terms of speed/effectiveness. – Vaidd4 Feb 22 at 10:35

Is June 2019 and that is no longer true in latest Google Chrome, the difference is now near non-existent (9:10) – Ivan Castellanos Jun 14 at 23:39

This appears the most performant of the ES6 variants: jsben.ch/HvICq – Brian M. Hunt Jul 5 at 19:22

Try

3

```
let swap = (o,r={})=> Object.keys(o).map(x=>r[o[x]]=x)&&r;
```

Show code snippet

edited Jan 15 at 23:22

answered Jan 15 at 22:50

Kamil Kiełczewski

21.9k 10 101 119

Using Ramda:

2

```
const swapKeysWithValues =
  R.pipe(
    R.keys,
    R.reduce((obj, k) => R.assoc(source[k], k, obj), {})
  );

const result = swapKeysWithValues(source);
```

edited Mar 24 '18 at 7:17

answered Oct 28 '17 at 14:18

im.pankratov

903 9 13

1

invert-kv: Invert the key/value of an object. Example: {foo: 'bar'} → {bar: 'foo'}

https://www.npmjs.com/package/invert-kv

```
const invertKv = require('invert-kv');

invertKv({foo: 'bar', unicorn: 'rainbow'});
//=> {bar: 'foo', rainbow: 'unicorn'}
```

answered Mar 6 at 13:46

Huan
820 11 23

Why is using yet another library better than a one-line solution like @Sc0ttyD suggests, or even a simple for loop? – Arel Jun 29 at 13:47

When we want more readability. ;-) – Huan Jun 30 at 19:18

1

With pure Ramda in a pure and point-free style:

```
const swapKeysAndValues = R.pipe(
   R.toPairs,
   R.map(R.reverse),
   R.fromPairs,
);
```

Or, with a little more convoluted ES6 version, still pure functional:

```
const swapKeysAndValues2 = obj => Object
    .entries(obj)
    .reduce((newObj, [key, value]) => ({...newObj, [value]: key}), {})
```

answered May 28 at 9:54

Pietro Bellone
11 1

0

```
var data = {A : 1, B : 2, C : 3, D : 4}
var newData = {};
Object.keys(data).forEach(function(key){newData[data[key]]=key});
console.log(newData);
```

edited Jun 11 at 10:13

answered Jun 10 at 9:06

Anup Agarwal

1 1

3 Where does this `object` come from? – Maxime Launois Jun 10 at 9:11

and this is not what `map` is supposed to do, you're using it here like `forEach` – barbsan Jun 10 at 10:24

Thank you for correcting me @MaximeLaunois – Anup Agarwal Jun 11 at 10:14

I never used forEach. I use underscore.js's each function so didn't knew about forEach. Thank you @barbsan – Anup Agarwal Jun 11 at 10:15

0

Here is a pure functional implementation of flipping `keys` and `values` in ES6:

TypeScript

```
  const flipKeyValues = (originalObj: {[key: string]: string}): {[key: string]: string}
=> {
    if(typeof originalObj === "object" && originalObj !== null ) {
      return Object
      .entries(originalObj)
      .reduce((
        acc: {[key: string]: string},
        [key, value]: [string, string],
      ) => {
        acc[value] = key
        return acc;
      }, {})
    } else {
      return {};
    }
  }
```

```
const flipKeyValues = (originalObj) => {
    if(typeof originalObj === "object" && originalObj !== null ) {
        return Object
        .entries(originalObj)
        .reduce((acc, [key, value]) => {
          acc[value] = key
          return acc;
        }, {})
    } else {
        return {};
    }
}

const obj = {foo: 'bar'}
console.log("ORIGINAL: ", obj)
console.log("FLIPPED: ", flipKeyValues(obj))
```

Run code snippet Expand snippet

edited Sep 17 at 0:08

answered Sep 5 at 17:47

Artur Grigio
1,848 2 26 49