How can you sort an array without mutating the original array?

Asked 9 years, 8 months ago Active 1 month ago Viewed 155k times



Let's suppose I wanted a sort function that returns a sorted copy of the inputted array. I naively tried this

354



31

```
function sort(arr) {
  return arr.sort();
}
```

()

and I tested it with this, which shows that my sort method is mutating the array.

```
var a = [2,3,7,5,3,7,1,3,4];
sort(a);
alert(a); //alerts "1,2,3,3,3,4,5,7,7"
```

I also tried this approach

```
function sort(arr) {
  return Array.prototype.sort(arr);
}
```

but it doesn't work at all.

Is there a straightforward way around this, prefereably a way that doesn't require hand-rolling my own sorting algorithm or copying every element of the array into a new one?

```
javascript arrays sorting
```

Share Improve this question Follow

```
edited Dec 23 '20 at 15:53
frederj
1,124 7 17
```

asked Mar 6 '12 at 22:11



Peter Olson

7k 47 194 236

- 1 create a deep copy of the array and sort it instead. evanmcdonnal Mar 6 '12 at 22:12
- 1 @evanmcdonnal A shallow copy might be good enough if all is wanted is a reordering and not a duplicate of every item in the array. − Kekoa Mar 6 '12 at 22:14 ✓

```
.sort requires the this value to be the array, so for the last snippet to work you would do .sort.call(arr) (though it doesn't solve your problem). – pimvdb Mar 6 '12 at 22:15
```

@Kekoa Yeah that's a good point. There is no need to consume more memory if you're only going to change the order of the elements and not the elements themselves. – evanmcdonnal Mar 6 '12 at 22:16

zzzzBov's method is working like a charm! <u>stackoverflow.com/a/9592774/7011860</u> – zimmerbimmer Jun 27 '19 at 6:14

6 Answers

Active Oldest Votes



You need to copy the array before you sort it. One way to make a **shallow copy** with es6:



```
const sorted = [...arr].sort();
```



The spread-syntax as array literal (copied from mdn):



```
var arr = [1, 2, 3];
var arr2 = [...arr]; // like arr.slice()
```

https://developer.mozilla.org/en-

<u>US/docs/Web/JavaScript/Reference/Operators/Spread_operator</u>

Share Improve this answer Follow



answered Feb 24 '17 at 15:52



this is really great.i think easier to understand than the concat and other approaches – sktguha Aug 31 '20 at 20:02

What do you mean with "not valid Javascript"? What is wrong or missing? – Putzi San Jun 3 at 15:03 🖍

this code does work, but when I compile the JS with Gulp it comes back with an error, SyntaxError: Unexpected token: punc (.) – brassmookie Jun 10 at 13:57

- To those saying it's not valid JavaScript... it's perfectly valid. If you're in Chrome/Safari/Edge or Firefox: open the dev console, define an array called arr and paste the expression to see the result.

 shangxiao Jun 17 at 7:48
- 2 @Cerin It sounds like you are on an incredibly outdated version of JS. Kloar Sep 10 at 15:31 🧪



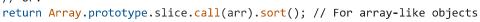
Just copy the array. There are many ways to do that:





```
function sort(arr) {
  return arr.concat().sort();
}
```





Share Improve this answer Follow

answered Mar 6 '12 at 22:13



Rob W **322k**

22k 76 771 656

Will this do a deep copy, i.e., will nested objects and arrays also be copied? — Peter Olson Mar 6 '12 at 22:15

Is there any advantage to using concat over say slice(0) or are they all pretty much just the same?

— JaredPar Mar 6 '12 at 22:15

- 4 @PeterOlson No, it's a shallow copy. If you really want a deep copy, use the search feature on Stack Overflow to find existing excellent answers for that. Rob W Mar 6 '12 at 22:19
- 15 Slice is now reported as notably faster Zander Brown May 31 '17 at 13:51
- 4 why Array.prototype.slice.call(arr).sort(); instead of arr.slice().sort(); ? Olivier Boissé Oct 25 '19 at 19:08



Try the following



```
function sortCopy(arr) {
  return arr.slice(0).sort();
}
```



The slice(0) expression creates a copy of the array starting at element 0.

Share Improve this answer Follow

answered Mar 6 '12 at 22:13



JaredPar **689k** 140 1199 1431



You can use slice with no arguments to copy an array:

47





Share Improve this answer Follow edited Mar 29 '16 at 0:19

answered Mar 6 '12 at 22:14



zzzzBov 163k 49 310 351

This answer is awesome! I'm surprised JavaScript allows mutation to this degree. Seems wrong. Thanks, again. – user3054109 Aug 25 '16 at 19:15



You can also do this

18

```
d = [20, 30, 10]
e = Array.from(d)
e.sort()
```



This way d will not get mutated.

```
function sorted(arr) {
  temp = Array.from(arr)
  return temp.sort()
}
```

```
//Use it like this
x = [20, 10, 100]
console.log(sorted(x))
```

Share Improve this answer Follow

answered Jan 7 '18 at 6:48

Aditya Agarwal

This answer is nice - Leasye Feb 18 '20 at 9:28



Anyone who wants to do a deep copy (e.g. if your array contains objects) can use:



let arrCopy = JSON.parse(JSON.stringify(arr))



Then you can sort arrcopy without changing arr.



arrCopy.sort((obj1, obj2) => obj1.id > obj2.id)

Please note: this can be slow for very large arrays.

Share Improve this answer Follow

answered Jan 21 '20 at 10:26

Hamada

39 1

- 1 This will work with instead of > in your second example. tkit May 15 '20 at 16:57
- and remember all your items should be serializable in order to bring them back after stringifying (eg. date objects, functions and symbols are problematic in this method) btargac Jan 20 at 13:47