How to copy array in javascript

Posted on May 4, 2019 | by Prashant Yadav

Posted in Arrays, Javascript

Learn how to copy an array in javascript.

There are two different types of copy that can be performed on an array.
1) Shallow Copy.
2) Deep Copy

Shallow copy an array

Arrays in javascript are just objects with some additional properties and methods which make them act like an array. So when we directly assign array to another variable it is shallowly copied.

This means it is just referencing to the array not copying all the elements.

```
let arr = [1, 2, 3, 4, 5];

//Shallow copy the array
let arr2 = arr;
console.log(arr2);
//[1, 2, 3, 4, 5];

//Update original array
arr.push(6);
//[1, 2, 3, 4, 5, 6]

//Update is reflected in the copied variable
console.log(arr2);
//[1, 2, 3, 4, 5, 6]
```

Deep copy an array

We will see different methods through which we can deep copy an array in javascript.

Practically prepare for your JavaScript interview

JavaScript Revision
JavaScript-Concept Based Problems
Data Structures
Algorithms
Machine Coding
Web Fundamentals

Using ES6 Spread Operator

We can use `...` operator to create a deep clone of a given array.

```
Copy
let arr = [1, 2, 3, 4, 5];

let arr2 = [...arr];
console.log(arr2);
//[1, 2, 3, 4, 5]

arr[1] = 10;
[1, 10, 3, 4, 5];


console.log(arr2);
//[1, 2, 3, 4, 5]
```

You can also use the `push()` method to create a deep clone.

```
Copy
let arr = [1, 2, 3, 4, 5];
let arr2 = [];

//Deep clone
arr2.push(...arr);

console.log(arr2);
//[1, 2, 3, 4, 5]
```

This method only works for literal values like Strings, Booleans, Numbers. It does not work for nested arrays, objects, and prototypes.

```
Copy
let arr = [{a: 1}, {b: 2}, {c: 3}];

let arr2 = [...arr];
//[{a: 1}, {b: 2}, {c: 3}];

arr[0].a = 10;
console.log(arr);
//[{a: 10}, {b: 2}, {c: 3}]

console.log(arr2);
//[{a: 10}, {b: 2}, {c: 3}];
```

By concatenating with empty array

We can concatenate the original array with an empty array and return it to make a deep clone.

```
let arr = [1, 2, 3, 4, 5];
let arr2 = [].concat(arr);

console.log(arr2);
//[1, 2, 3, 4, 5]
```

Just like the spread operator, it will only work for literal values only.

```
let arr = [{a: 1}, {b: 2}, {c: 3}];

let arr2 = [].concat(arr);
//[{a: 1}, {b: 2}, {c: 3}];

arr[0].a = 10;
console.log(arr);
//[{a: 10}, {b: 2}, {c: 3}]

console.log(arr2);
//[{a: 10}, {b: 2}, {c: 3}];
```

Just like the above options, we can also use Array.from(), Array.splice() and Array.slice() methods to achieve the same but all of these only deep copy literal values.

Using JSON.parse() and JSON.stringify()

`JSON.parse()` converts a string to JSON and `JSON.stringify()` converts a JSON to a string.

We can combine them together to create a deep copy of nested arrays and arrays of objects.

```
let arr = [{a: 1}, {b: 2}, [1, 2]];

let arr2 = JSON.parse(JSON.stringify(arr));

console.log(arr2);
//[{a: 1}, {b: 2}, [1, 2]];

arr[0].a = 10;
console.log(arr);
//[{a: 10}, {b: 2}, [1, 2]];

console.log(arr2);
//[{a: 1}, {b: 2}, [1, 2]];
```

This method works great but you should be really careful about the data that can be stored in JSON.

```
let arr = [1, undefined, 2];
let arr2 = JSON.parse(JSON.stringify(arr));
// undefineds are converted to nulls
console.log(arr2);
//[1, null, 2]


let arr = [document.body, document.querySelector('p')];
let arr2 = JSON.parse(JSON.stringify(arr));
// DOM nodes are converted to empty objects
console.log(arr2);
//[{}, {}]


let arr = [new Date()];
let arr2 = JSON.parse(JSON.stringify(arr));
// JS dates are converted to strings
console.log(arr2);
//["2019-05-04T10:26:01.390Z"]
```

Also, this cannot be used to make a complete deep copy, it will still not work for prototypes.

Using $.extend() of Jquery

`$.extend(deep, copyTo, copyFrom)` can be used to make a complete deep copy of any array or object in javascript.

```
let arr = [{a: 1}, {b: 2}, [1, 2]];

//Deep copy
let arr2 = $.extend(true, [], arr);

arr[2][0] = 5;

console.log(arr2);
//[{a: 1}, {b: 2}, [1, 2]];

console.log(arr);
//[{a: 1}, {b: 2}, [5, 2]];
```

You can also use other libraries like lodash, underscore to achieve the same.

Vanilla Javascript

You can also create a custom function to create a deep clone of objects or arrays in vanilla javascript.

Copy

```
let deepCopy = (aObject) => {
  //If not a object then return
  if (!aObject) {
    return aObject;
  }

  let v;

  //Check the type of the input
  let bObject = Array.isArray(aObject) ? [] : {};

  //Copy each element
  for (const k in aObject) {
    v = aObject[k];

    //If type of element is object
    //Then recursively call the same function and create  a copy
    bObject[k] = (typeof v === "object") ? deepCopy(v) : v;
  }

  return bObject;
}
```

Copy

```
let arr = [{a: 1}, {b: 2}, [1, 2]];

//Create a deep copy
let arr2 = deepCopy(arr);

//Update the original array
arr[2][0] = 10;

console.log(arr2);
//[{a: 1}, {b: 2}, [1, 2]]
```

Prepare for your **JavaScript Interview** practically on each Interview rounds and grab that job.

BEGIN LEARNING

Recommended Posts:

- Javascript const vs var
- Remove cycle from the object in JavaScript
- useScript() hook in React
- How to open new tab in javascript
- Get the class name of the object in JavaScript
- Number of subarrays with given sum k

[Create a basic implementation of a streams API]
[Implement getByClassNameHierarchy() function in JavaScript]
[usePrevious() hook in React.]
[Convert a string to uppercase in javascript]

Prev Next