How do you deep copy an array with JavaScript?

Posted At : December 11, 2018 9:00 AM | Posted By : Jeffry Houser
Related Categories: JavaScript, Professional, Angular, TypeScript

The Problem

I was working on an Angular component that accepted an array input and created a drop down from it. As part of our internal architecture the component would add a disabled property to certain items on the input. This would permanently modify the input array, which was cached by the app and used elsewhere. We couldn't have that. We decided to make an internal copy of the array so that our local component changes would not affect the global data store. How do we do that?

Normally when I want to create a deep copy of an object, I turn to Try it here

```
const object1 = [
  {value: 1},
  {value: 2},
  {value: 3}
];

const object2 = Object.assign({}, object1);

console.log(object1);

console.log(object2);
```

Take a look at the output:

If you look closely at the output between object 1 and object 2, you'll discover that object2 is no longer an array, it is an object. That was the wrong syntax and not what we intended.

Object.Assign() with an Array

We could use Object.assign() with an array instead of an object, like this:

```
const object3 = Object.assign([], object1);
```

That will give us an array. Make sure that they are separate objects:

```
object1.push({value:4})
console.log(object1);
console.log(object3);
```

That does solve the issue and we have an array:

It isn't the only way to solve the issue, though.

Use a spread operator

The spread operator will do the same thing.

```
const object4 = [...object1];
object1.push({value:4})
console.log(object1);
console.log(object4);
```

I always forget about the spread operator, but it works well. Depending where you look, there are concerns about it's performance over other options. I'm reusing the same screenshot because there is no change:

Use Slice

You can also create a deep copy of an array using array.slice():

```
const object5 = object1.slice();

object1.push({value:4})

console.log(object1);
console.log(object5);
```

Repeated screenshot because nothing changed:

This is the way I went with, for no other reason than it was used elsewhere in the code base and it pays to be consistent.

Hopefully you learned something. ;)