Clone Arrays with JavaScript

By David Walsh on July 14, 2012 38

Believe it or not, there are reasons we use JavaScript frameworks outside of animations and those sexy accordions that people can't do without. The further you get into high-powered JavaScript applications (assuming you're creating true web *applications*, not *websites*), the more the need for basic JavaScript functionalities; i.e. JavaScript utilities that have nothing to do with DOM. One of those basic utilities is the ability to clone an array. Quite often I see developers iterating over array items to create their clone; in reality, cloning an array can be as easy as a `slice` !

The JavaScript

To clone the contents of a given array, all you need to do is call `slice` , providing `0` as the first argument:

```
var clone = myArray.slice(0);
```

The code above creates clone of the original array; keep in mind that if objects exist in your array, the references are kept; i.e. the code above does not do a "deep" clone of the array contents. To add `clone` as a native method to arrays, you'd do something like this:

```
Array.prototype.clone = function() {
    return this.slice(0);
};
```

And there you have it! Don't iterate over arrays to clone them if all you need is a naive clone!

Recent Features

CSS Filters

CSS filter support recently landed within WebKit nightlies. CSS filters provide a method for modifying the rendering of a basic DOM element, image, or video. CSS filters allow for blurring, warping, and modifying the color intensity of elements. Let's have...

Vibration API

Many of the new APIs provided to us by browser vendors are more targeted toward the mobile user than the desktop user. One of those simple APIs the Vibration API. The Vibration API allows developers to direct the device, using JavaScript, to vibrate in...

Incredible Demos

AJAX Page Loads Using MooTools Fx.Explode

Note: All credit for Fx.Explode goes to Jan Kassens. One of the awesome pieces of code in MooTools Core Developer Jan Kassens' sandbox is his Fx.Explode functionality. When you click on any of the designated Fx.Explode elements, the elements "explode" off of the...

Fixing sIFR Printing with CSS and MooTools

While I'm not a huge sIFR advocate I can understand its allure. A customer recently asked us to implement sIFR on their website but I ran into a problem: the sIFR headings wouldn't print because they were Flash objects. Here's how to fix...

Discussion

faruzzy

This is a great tip! so worth it!

Michele

Interesting tip!

Robert Schultz

Just to note, I use `.slice()` all the time without passing in 'zero' and have never seen an issue. So even though the MDN docs do not say it is optional, it appears to work just fine with no arguments :)

Tom Lo

If you pass Zero as argument is run faster:
http://jsperf.com/new-array-vs-splice-vs-slice/19

Mark

It is also much faster, especially for large arrays. Here's a benchmark I put together:

http://jsperf.com/loop-vs-slice-copy/3

mesuutt

What is the differences between `clonearr=originalarr` and `clonearr=originalarr.slice(0)` ?

jonathan de montalembert

I don't understand, why not use clone = myArray?

RevMen

`clone = myArray` makes clone a reference to `myArray` , not a copy of it.

Farzad YZ

Arrays are a specific type of objects. as objects are being passed by reference; meaning that changing source object will change result object and vice versa; you have to clone the result not copy it.

Stefan

Mesuutt and Jonathan: doing `cloneArr = originalArr` will only copy the pointer of the array and not clone it:

```
var a = [1,2,3]; // a.length == 3
b = a; // a.length == 3 and b.length == 3
b.push(4); // a.length == 4 and b.length == 4
```

mesuutt

thanks @stefan, I understand exactly now :)

John J. Camilleri

There's a subtle bug that can result from adding something to the prototype of Array. If you iterate over arrays with using `for...in` construct:

```
var a = [3,7,9];
for (var i in a) {
  console.log(i);
}
```

Note that the for loop will also iterate over the case when i == "clone"!
To be safe, you must iterate by using the length property:

```
for (var i=0; i<a.length; i++) {
  console.log(i);
}
```

Farzad YZ

This is because of

```
for..in
```

construct not the Array prototype.

Eric H.

Take note: if your array is filled with objects, those objects remain linked:

```
var aAll = [{x:0, y:0}, {x:1, y:1}],
    aAll1 =aAll.slice(0);
console.log(aAll[0].x, aAll1[0].x); // 0, 0
aAll1[0].x += 10
console.log(aAll[0].x, aAll1[0].x); // 10, 10
aAll1[0] = {};
console.log(aAll[0].x, aAll1[0].x); // 10, undefined
```

MikeOne

So what if I need to create a clone of an array filled with objects? I cannot have the objects to be by reference in the clone. Iteration my only option?

Scott S. McCoy

No one noticed this empties the array in question? That's not exactly a clone.

Scott S. McCoy

It's really lame you cannot delete or edit comments here.

Use `slice()` not `splice()`.

JC

That's good that you cannot delete your comment! I did the same error and I realized what was going on only by reading your comment... :)

Kadimi

Maybe it's worth mentioning in a comment, this is very similar to how Python copies lists (arrays)

```
my_list = [1, 2, "some text"];

// To create a refrence
my_ref = my_list

// To create a copy
my_copy = my_list[:]
```

Etai

Adding Array.prototype.clone as mentioned in this article is a terrible idea... as it's only a shallow clone, and if another developer on your team will use your clone method, they can trip up over this.

Stepan Suvorov

Hi, thx for solution, exactly what Iwas looking for

Felipe N. Moura

Hi.
I know this post is kinda old, but we were on a discussion about it around here, and we prepared these tests.
http://jsperf.com/array-content-copy/2
many interesting thing around here!
for instance, array.slice is the fastest on google chrome, while in firefox, array.concat was the fastest.
Another interesting thing is the way the browser optimizes the most verbose test, which is not the slowest one!

sean

a handy way to get a deep copy is:

```
var clone = JSON.parse( JSON.stringify( myArray ) );
```

Farzad YZ

JSON will do the job most of the time but would fail when array contains another json or a function.

Felipe N Moura

I added this method, using JSON's `parse` / `stringify` methods to that perf.
It works, but this does not have the best of the performances:

http://jsperf.com/array-content-copy/7

Massimo

```
var clone = myArray.slice(0);
```

don’t make a clone. Is the same of

```
clone=myArray;
```

every change on clone is reflected on myArray. To make a clone you must copy every element singly.

sean

not according to this test

```

// bad clone
var a = [1,2,3,4,5];
var b = a;
a.push(99);
console.log(a); // [1,2,3,4,5,99]
console.log(b); // [1,2,3,4,5,99]

// good clone
var x = [1,2,3,4,5];
var y = x.slice(0);
x.push(77);
console.log(x); // [1,2,3,4,5,77]
console.log(y); // [1,2,3,4,5]
```

Felipe Moura

Yep, that’s not true, Massimo.
try this out

```
var a1= [1, 2, 3, 4];
var a2= a1.slice(0);
a2[2]= 'X';
console.log(a1); // [1, 2, 3, 4]
console.log(a2); // [1, 2, "X", 4]
```

Andy

This will only work if your array has simple values in it. If your array contains objects (or other arrays!!!) it will not copy them.

For example

```
 var a = [ [ 1 ] ];
var b = a.slice( 0 );
a[0] === b[0] // true!
```

Be careful! here is a function you can use if you know your arrays are nested:
http://blog.andrewray.me/how-to-clone-a-nested-array-in-javascript/

Felipe N. Moura

Hi Andy.

That's not entirely the case!
What happens here, is that objects and arrays are passed as reference, and even though, what you were comparing there was their value at [0]!
If you simply run this, you will understand:

```
a[0]= 'x';
console.log(b[0]); // [1]
```

By changing a[0], you are NOT altering b[0].
BUT, if you had done this, instead:

```
a[0][0] = 'a';
console.log(b[0][0]); // "a"
```

This time it changed the array on "b", but because it was a reference to the same array/object, therefore, any change into one, will be applied to the other.

So, yes, we must be always aware of objects being passed from some place to another, in any case :)
In the example you gave, b is a copy of a, but as a copy, it also copied the pointers/references to the original Array stored on each position.

Space1000

Man you're cool! Can I omit '0' in .slice(0)?

Taner

`slice()` function is working only numeric indexed arrays, not working this array:

```
var a = [];
a["a"] = 1; a["b"] = 2; a["c"] = 3;
```

This version is working with all arrays :

```
Array.prototype.clone = function() {
    var clone = [];
    for(var i in this)
    {
        clone[i] = this[i]
    }
    return clone;
};
```

Taner

Sean's code more efficiently:

```
var clone = JSON.parse( JSON.stringify( myArray ) );
```

This clones also object elements in the array

Erik

The last time I tested stringify for cloning it was a lot slower than creating objects and copying values over.

Vince

More efficiently ? This is so bad in term of performance.

Erik D

Do not post nonsense on the web David Walsh.
`slice()` is not a clone method at all!

Slice is a copy method! In other words it copies the values/references from one array to another.

So when you copy objects from array A to B and modify a single item that is referenced in both arrays. Both array A and B will display the new values for this item.
Cloning has a different meaning.
To actually clone objects you must create new objects and copy over values.

Vince

"Don't iterate over arrays to clone them if all you need is a naive clone!"

This is what slice does internally -_- and Erik is true, it's inappropriate to talk about clone there.

June Rockwell

Is there a significant difference between `.slice()` and `.slice(0)`? I've always used the latter to deep copy. What's the significance of the first argument being `0`? And, it's not necessary, is it?

Name | Email | Website

Wrap your code in `<pre class="{language}"></pre>` tags, link to a GitHub gist, JSFiddle fiddle, or CodePen pen to embed!

Continue this conversation via email

Post Comment! | Use Code Editor