Empty an Array with JavaScript

By David Walsh on December 24, 2012 24

Emptying an array is a common JavaScript task but too often I see the task performed in the incorrect way. Many times developers will create a new array:


```
myArray = []; // bad
```

That isn't the optimal way to get a fresh array; to truncate an array, and thus empty it, you should set the length property to zero:

```
myArray.length = 0; // good!
```

Setting the length equal to zero empties the existing array, not creating another array! This helps you to avoid pointer issues with arrays as well. With the new array method above:

```
// ... as provided by Dag in the comments
A = [1,2,3,4,5]
B = A
A = []
console.log(B) // [1,2,3,4,5]
```

The `length` truncation would make both `A` and `B` the original array but empty!

Recent Features

How to Create a Twitter Card

One of my favorite social APIs was the Open Graph API adopted by Facebook. Adding just a few META tags to each page allowed links to my article to be styled and presented the way I wanted them to, giving me a bit of control...

5 HTML5 APIs You Didn't Know Existed

When you say or read "HTML5", you half expect exotic dancers and unicorns to walk into the room to the tune of "I'm Sexy and I Know It." Can you blame us though? We watched the fundamental APIs stagnate for so long that a basic feature...

Incredible Demos

Create a 3D Panorama Image with A-Frame

In the five years I've been at Mozilla I've seen some awesome projects. Some of them very popular, some of them very niche, but none of them has inspired me the way the MozVR team's work with WebVR and A-Frame project have. A-Frame is a community project...

iPhone Checkboxes Using MooTools

One of the sweet user interface enhancements provided by Apple's iPhone is their checkbox-slider functionality. Thomas Reynolds recently released a jQuery plugin that allows you to make your checkboxes look like iPhone sliders. Here's how to implement that functionality using the beloved...

Discussion

jhuesos

I am guilty in this one. But, anybody could add what are the cons of using the anti-pattern?

I guess performance because you don't have to instantiate a new object. But how the gc behaves for both approaches?

Cheers

Kris Zyp

I didn't create my own, here is a jsperf test that seems to suggest otherwise:
http://jsperf.com/array-destroy
I'd be curious if there are situations where a.length = 0; is faster (and how many iterations are necessary to compensate for the extra bytes of download time).

Kyle Simpson

This is a really bad test. It's not going to give anywhere near reliable results.

Asher

Based on Kris' actual test, it seems like `a.length = 0` is worse on both readability AND performance. I would venture that most dev's recognize `a = []` at a glance, but `a.length = 0` would take that much more thinking.

Kyle Simpson

The test is entirely bogus. Don't base your conclusions on bad benchmarks.

Brandon

While it does prevent allocation of a new object, in order to empty the array all the indices in it must be deleted (that's what setting the length to 0 does). This operation may or may not be optimized, depending on how a particular js engine works.

seelts

«That isn't the optimal way»

Why?

Dag

I guess it is because A = [] actually does not empty the Array, it only changes the pointer.

Example:

```
A = [1,2,3,4,5]
B = A
A = []
console.log(B) // [1,2,3,4,5]
```

Rocka84

I have to contradict here. As other comments stated before, the speed of the array.length approach isn't better (if not worse) than just creating a new array and with decent garbage collectors, memory shouldn't be a problem either.
And even if the length property *is* writable, I personally think it *should* be treated as read-only.

But feel free to convince me of your way, if you can ;-)

David Walsh

My thought was avoiding reliance on garbage collection and avoiding pointer issues. Creating a new array may be more important but it does come with caveats that people need to be aware of.

Maksim

Isn't it default js object behavior though?

```
var a = {a:'a'};
var b = a;
a = {};
console.log(b); // {a:'a'}
```

Jeff Griffiths

s/exiting/existing/

Jay

What about doing `delete myArray;` then?

Paul Comanici (darkyndy)

I think that a lot can be discussed on this subject without having relevant data.

I've created a test script, source code: https://gist.github.com/4420845

```
var myArray,
  myInitialArray,
        cleanArray,
        continueLoopTest;

myInitialArray = [1,2,3,4,5,6];
myInitialArray = [{a: 1, b: "2"},{a: 1, b: "2"},{a: 1, b: "2"},{a: 1, b: "2"},{a

cleanArray = {
        byLength: {
                getTitle: function () {
                        return "clean array by setting length to 0";
                },
                clean: function () {
                        myArray.length = 0;
                }
        },
        byNew: {
                getTitle: function () {
                        return "clean array by setting array to []";
                },
                clean: function () {
                        myArray = [];
                }
        }
};

function runTest() {
        var pageTitleEl,
                useMethod;
        useMethod = "byNew";
        //useMethod = "byLength";

        pageTitleEl = document.getElementsByTagName("title")[0];
        pageTitleEl.innerText = cleanArray[useMethod].getTitle();

        function executeTest() {
                var iterationNr = 0,
                        endTest;
                endTest = Date.now() + 1000;
```

```
                while (endTest > Date.now()) {
                        myArray = myInitialArray;
                        cleanArray[useMethod].clean();
                        iterationNr += 1;
                }
                console.log(iterationNr);
        }
        executeTest();
}

setTimeout(runTest, 100);
```

empty-array.js hosted with ♥ by GitHub view raw

So we have (for reference):
“A” – clear by using length;
“B” – clear by using [];

I did tests only in Google Chrome v23 on Windows 7.
As operations per second case “B” is much faster (almost +1 million ops per sec);

My test script loops for one second, what I noticed with case “B” is that memory is cleared faster then with method “A”. There wasn’t any memory leak (in both cases);

@Dag & @David Walsh:
regarding:

```
A = [1,2,3,4,5]
B = A
A = []
console.log(B) // [1,2,3,4,5]
```

I tried to find in ES5 specs how this should work, but couldn’t find now, what I remember from there is that objects (arrays also apply) are set by reference
`B = A`
B points to the same data
when you assign something to A then A will not have any more the reference to old array object, but B will continue to have data that was before on A.

Another thing to keep in mind with ES5 that by default length property on Array has “writable” set to true, if you will set it to false this method will not work any more.

I will stick with: `myArr = [];` as I have better performance and I don't see anything wrong with it. Still if I'm missing something please post.

@David Walsh:
For the part "Creating a new array may be more important but it does come with caveats that people need to be aware of.", can you explain what are the downsides?

Markus

I suppose that adding elements into an array which was cleared using .length = 0 is faster, because the memory is already allocated and it is less likeley that the js engine has to enlarge the array at runtime (which is a slow operation in other languages, dont know if it is also the case in JS).

JuanO

I'm sure that wrapping all the possible logic for clearing an array within a function would add some time to your measurements. In the end though, commented code will help with unfamiliar concepts and if .length is not writeable, well, we can handle that within the wrapper.

John R Pittman

These are my first days at giving back to the open source community and will be going by the handle TreeTopAnonymous on github as well as anywhere else that's cool, so I hope this is helpful. I'm just starting my path into hardcore javascript development and love it so I figured I might as well optimize some techniques for my adventures.

Test cases: http://jsperf.com/array-clearing-performance

You can always loop through the array popping each object off which changes the actual array size as well and will allow you to remain in contain with the original array reference without having to instantiate a new array object. I do use shift() over pop() for this due to shift() being faster.

During my tests I noticed length = 0; method starts winning at arrays >= 15000 but gets destroyed in smaller index counts so I created 3 different tests to show scaling.

In the end my way for looping and shifting the array wins this performance battle overall unless you have the need for clearing extremely large arrays which then you could add a logic check for at over a certain amount to use the length = 0; method; then you will need to understand on faster hardware the number of 15000 will probably increase.

John R Pittman

Update to previous post: Well it seems now that no matter the scaling, my shift() way beats the pants off over them both. Not sure what I was looking at before.

prithvi

i am agree with Dog
because A = [] actually does not empty the Array, it only changes the pointer.

Kyle Simpson

I think this is a much more accurate performance test:

http://jsperf.com/truncating-arrays-correctly

Comments welcome if I missed anything there.

TL;DR: the *speed* performance diffs are pretty small, and pretty irrelevant. Unquestionably, though, there will be a memory diff between `= []` and `.length = 0`. If you care to reduce memory churn (GC), especially on mobile devices, `.length = 0` is a better way to go.

Ben West

Please update this post to indicate whether all browsers (particularly legacy ones like IE7 or earlier) and execjs implementations are able to detect that the pointers to the items referenced by your array are no-longer 'referenced' as the array still points to them, even though they aren't accessible via other methods: for-loops, `forEach`, etc.

Also, I'd like to question when you would do this and whether is optimization isn't universally premature, as I don't know that I have ever assigned an empty array, except to initialize a variable for the first time.

Cristian Elena

I use:

```
while(a.length){
    a.pop();
}
```

Patrick

It seems obvious that clearing the array either by popping all elements or setting length to zero is going to be slower. This approach requires you to synchronously empty out the underlying data from the array. In contrast, deleting, nulling, or replacing with a new empty array, defers the “real” cleanup. The original reference is placed to the side, and the engine can tackle actually doing something with that old array when it needs the memory.

I wouldn’t have gone as far as to say that constructing a new array is bad, but rather that it’s a potential gotcha. If your code has multiple references to the same array, every time you clear the array destructively, you’ll need to update all those references.

gghez

https://jsperf.com/array-destroy

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