Most popular (last 30 days)

ECMAScript 2017: the final feature set

ECMAScript 6 modules: the final syntax

ES proposal: import() – dynamically importing ES modules

Making transpiled ES modules more spec-compliant

ES proposal: Shared memory and atomics

Classes in ECMAScript 6 (final semantics)

Communicating between Web Workers via MessageChannel

Most popular (all time)

ECMAScript 6 modules: the final syntax

Classes in ECMAScript 6 (final semantics)

Iterating over arrays and objects in JavaScript

ECMAScript 6's new array methods

The final feature set of ECMAScript 2016 (ES7)

WebAssembly: a binary format for the web

Basic JavaScript for the impatient programmer

Google Dart to "ultimately ... replace JavaScript"

Six nifty ES6 tricks

Google's Polymer and the future of web UI frameworks

Blog archive

► 2017 (4)

► 2016 (38)

2011-08-01

JavaScript performance: Array.prototype versus []

Labels: dev, javascript, jslang

`Array.prototype` contains many *generic methods* that can be applied to *array-like* objects. [] is a popular shortcut for accessing these methods. This post examines the pros and cons of using that shortcut.

Update: Inspired by a comment from Kevin Roberts, I've added a third way of accessing generic methods, and a conclusion.

Explanations

Array-like objects. Some objects in JavaScript are *array-like*, they have indexed access and a `length` property like arrays, but none of the array methods. Array-like objects include the special variable `arguments` (giving indexed access to all arguments that were passed to a function) and most DOM results. Not having the standard array methods is especially unfortunate under ECMAScript 5, which has goodies such as `Array.prototype.forEach`.

Generic methods. Some methods are *generic*. While they are directly available to instances of their prototype, they can also be borrowed by other instances. To borrow a generic method, one invokes one of the following two methods on it:

- `Function.prototype.call(thisValue, [arg1], [arg2], ...)`
- `Function.prototype.apply(thisValue, [arrayWithArguments])`

The borrowing instance is the first argument and becomes the value of `this`. Generic methods have to be written so that they require `this` to only have a minimal set of methods. For example, most generic array methods only need `this` to provide `length` and indexed access. `Array.prototype.slice` is generic and allows one to turn any part of an array-like object into an array.

Example: invoking `Array.prototype.map()` generically, on the array-like `arguments` object.

```
function prefixHello(prefix) {
    return Array.prototype.map.call(arguments, function(elem) {
        return "Hello "+elem;
    });
}
```

Interaction:

```
> prefixHello("Jane", "John")
[ 'Hello Jane', 'Hello John' ]
```

[] as a shortcut. `[].foo` is often used as a shortcut for `Array.prototype.foo`. That is, you access a prototype property via an instance.

- Pro: More compact.
- Con: Does not really describe one's intent. You are not trying to invoke an instance method, you are borrowing a function from the prototype.
- Con: Slightly slower (see below).

Dr. Axel Rauschmayer

Free online books by Axel

Speaking JavaScript [up to ES5]

Exploring ES6

JavaScript training: Ecmanauten

Timing several ways of accessing generic methods

I wanted to see how bad performance really suffered and did a quick non-scientific test. Test framework:

```
    var iterations = 100000000;
    var data = []; // empty so that slice() doesn't have much to do
    (function () {
        var start = (new Date).getTime();

        // loop

        var diff = (new Date).getTime() - start;
        console.log(diff);
    }());
```

Timing prototype access:

```
    for(var i=0; i<iterations; i++) {
        Array.prototype.slice.call(data);
    }
```

Timing the shortcut:

```
    for(var i=0; i<iterations; i++) {
        [].slice.call(data);
    }
```

Storing the prototype in a local variable:

```
    var arrayProto = Array.prototype;
    for(var i=0; i<iterations; i++) {
        arrayProto.slice.call(data);
    }
```

Results (iMac, 2.7 GHz Intel Core i5):

	iterations	prototype	shortcut	quick prototype
Node.js 0.4.8	100,000,000	5019ms	5075ms	4692ms
Firefox 6	10,000,000	1592ms	2237ms	1522ms
Rhino 1.7 release 3	10,000,000	2318ms	2687ms	1878ms

Conclusion

The time differences are not exactly earth-shattering. Thus, unless you are working with performance-critical code, you should choose whatever you think *reads* best (as opposed to what is easiest to write).

Tweets by @rauschma

Axel Rauschm... @rauschma
Enjoying "Troll Hunters":
– "Let's call him 'Gnome Chomsky'"
– "Juliet dies in this? Nooo!"
10h

Axel Rauschmayer Retweeted
Jonathan Creamer @jcreamer898
A nice little shout out to @rauschma... infoworld.com/article/316483 ... #es2017 #async
JavaScr...
ECMASc...
infoworld...
14h

Axel Rauschmayer @rauschma
Not a physics book!
twitter.com/ManningBooks/s ...
02 Feb

Axel Rauschmayer Retweeted
Jordan Harband @ljharb
Making our React components forbid extra props has caught SO many bugs. I highly recommend it.

npmjs.com/airbnb-prop-ty...
npm: air...
Custom ...
npmjs.com
02 Feb

Axel Rauschmayer Retweeted
Seth Petry-Johnson @spetryjohnson
Open strong. Be bold. Tell a story. Do something to get me interested. Opening w/ the "obligatory 'about me' slide" puts me to sleep :)
02 Feb

Google

Preventing the next Breivik

What's next for our economy?

iCloud icon versus classic Mac icon

Adobe Edge: a Flash-like tool for HTML5

JavaScript performance: Array.prototype versus []

Water reservoir in space: 140 trillion times our o…

Eight words not to use in your Twitter bio

► July (18)

► June (29)

► May (23)

► April (33)

► March (40)

► February (47)

► January (77)

► 2010 (174)

► 2009 (68)

► 2008 (46)

► 2007 (12)

► 2006 (1)

► 2005 (2)

Labels

dev (592)
javascript (400)
computers (316)
life (194)
jslang (179)
esnext (156)
apple (107)
webdev (95)
mobile (83)
scitech (50)
hack (49)
mac (47)
google (39)
java (37)
ios (33)
business (32)
video (32)
clientjs (31)
hci (27)
entertainment (26)
nodejs (26)
society (26)

6 Comments **The 2ality blog** 1 **Login**

Recommend **Share** **Sort by Best**

Join the discussion…

Yan Yankowski • a year ago

A good performance test https://jsperf.com/array-proto...

Reply • Share ›

Mathias Bynens • 3 years ago

There are several issues in the code you use to benchmark here. See http://mathiasbynens.be/notes/... for details. Why not create a jsPerf test instead?

Reply • Share ›

kevnroberts • 6 years ago

Did you try timing the shortcut without creating a new array in each iteration? Like so:

```
(function () {
    var start = (new Date).getTime()
        a = [];
    for(var i=0; i<iterations; i++) {
        a.slice.call(data);
    }
    var diff = (new Date).getTime() - start;
    console.log(diff);
}());
```

Reply • Share ›

Axel Rauschmayer Mod ➦ kevnroberts • 6 years ago

Good idea. But a = Array.prototype makes more sense. I've updated the post accordingly.

Reply • Share ›

kevnroberts ➦ Axel Rauschmayer • 6 years ago

I was just wondering if the performance difference between the Prototype and the instance of Array (via []) was in the fact that you create a new array in each iteration. So at the end of the function, you have
100,000,000 anonymous empty arrays hanging around.

I'm just curious if you used the same array object and used "slice()" from that one object instead of creating a new one every time, if that would increase performance or not.

Reply • Share ›

Jason Bunting • 6 years ago

"Slightly slower?" I'd say!

Reply • Share ›

Subscribe **Add Disqus to your site Add Disqus Add** **Privacy**

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

browser (25)

firefox (25)

html5 (24)

ipad (24)

movie (23)

psychology (22)

2ality (18)

tv (18)

android (17)

social (17)

chrome (16)

fun (16)

jsmodules (16)

tablet (16)

humor (15)

politics (15)

web (15)

cloud (14)

hardware (14)

microsoft (14)

software engineering (13)

blogging (12)

gaming (12)

eclipse (11)

gwt (11)

numbers (11)

programming languages (11)

app store (10)

media (10)

nature (10)

security (10)

semantic web (10)

software (10)

twitter (10)

webos (10)

12quirks (9)

education (9)

jstools (9)

photo (9)

webcomponents (9)

windows 8 (9)

async (8)

idea (8)

iphone (8)

itunes (8)

scifi-fantasy (8)

app (7)

babel (7)

bookmarklet (7)

chromeos (7)

english (7)

es proposal (7)

fringe (7)

html (7)

jsint (7)

jsshell (7)

thunderbolt (7)

webapp (7)

advancedjs (6)

blogger (6)

crowdsourcing (6)

latex (6)

lion (6)

promises (6)

ted (6)

book (5)

environment (5)

gadget (5)

googleio (5)

intel (5)

jsarrays (5)

jshistory (5)

layout (5)

light peak (5)

michael j. fox (5)

music (5)

pdf (5)

polymer (5)

shell (5)

tc39 (5)

template literals (5)

underscorejs (5)

vlc (5)

__proto__ (4)

coffeescript (4)

concurrency (4)

dart (4)

facebook (4)

gimp (4)

googleplus (4)

health (4)

howto (4)

hp (4)

javafx (4)

kindle (4)

leopard (4)

macbook (4)

motorola (4)

münchen (4)

occupy (4)

pl fundamentals (4)

presenting (4)

publishing (4)

series (4)

textbook (4)

web design (4)

amazon (3)

asmjs (3)
back to the future (3)
bitwise_ops (3)
css (3)
es2016 (3)
flattr (3)
fluentconf (3)
food (3)
foreign languages (3)
house (3)
icloud (3)
info mgmt (3)
jsfuture (3)
jsstyle (3)
linux (3)
mozilla (3)
python (3)
regexp (3)
samsung (3)
tizen (3)
traffic (3)
typedjs (3)
unix (3)
adobe (2)
angry birds (2)
angularjs (2)
astronomy (2)
audio (2)
comic (2)
design (2)
dom (2)
ecommerce (2)
eval (2)
exploring es6 (2)
facebook flow (2)
facets (2)
flash (2)
free (2)
futurama (2)
guide (2)
history (2)
hyena (2)
internet explorer (2)
iteration (2)
journalism (2)
jquery (2)
jsengine (2)
jslib (2)
law (2)
lightning (2)
markdown (2)
math (2)
meego (2)
month (2)

nike (2)

nokia (2)

npm (2)

programming (2)

raffle (2)

repl (2)

servo (2)

sponsor (2)

steve jobs (2)

travel (2)

typescript (2)

usb (2)

winphone (2)

wwdc (2)

airbender (1)

amdefine (1)

aol (1)

app urls (1)

architecture (1)

atscript (1)

basic income (1)

biology (1)

blink (1)

bluetooth (1)

canada (1)

clip (1)

coding (1)

community (1)

cross-platform (1)

deutsch (1)

diaspora (1)

distributed-social-network (1)

dsl (1)

dvd (1)

dzone (1)

emacs (1)

emberjs (1)

energy (1)

esnext news (1)

esprop (1)

example (1)

facetator (1)

feedback (1)

firefly (1)

firefoxos (1)

fritzbox (1)

german (1)

git (1)

guest (1)

guice (1)

h.264 (1)

home entertainment (1)

hosting (1)

htc (1)

- ical (1)
- jsdom (1)
- jsmyth (1)
- library (1)
- location (1)
- marketing (1)
- mars (1)
- meta-data (1)
- middle east (1)
- mpaa (1)
- msl (1)
- mssurface (1)
- netflix (1)
- nsa (1)
- obama (1)
- openoffice (1)
- opinion (1)
- oracle (1)
- organizing (1)
- philosophy (1)
- pixar (1)
- pnacl (1)
- prism (1)
- privacy (1)
- proxies (1)
- puzzle (1)
- raspberry pi (1)
- read (1)
- rodney (1)
- rust (1)
- safari (1)
- sponsoring (1)
- star trek (1)
- static generation (1)
- talk (1)
- technique (1)
- theora (1)
- thunderbird (1)
- typography (1)
- unicode (1)
- v8 (1)
- voice control (1)
- webassembly (1)
- webkit (1)
- webm (1)
- webpack (1)
- yahoo (1)