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)

2012-06-13

JavaScript: sparse arrays vs. dense arrays

Labels: dev, javascript, jsarrays, jslang, underscorejs

In general, arrays in JavaScript are *sparse* – they can have holes in them, because an array is simply a map from indices to values. This blog post explains how to create *dense* arrays, arrays without holes.

1. Sparse arrays

Creating a sparse array of a given length is simple:

```
> var a = new Array(3);
> a
[ , , ]
> a.length
3
> a[0]
undefined
```

When you iterate over it, you can see that it has no elements. JavaScript skips the holes.

```
> a.forEach(function (x, i) { console.log(i+". "+x) });

> a.map(function (x, i) { return i })
[ , , ]
```

2. Dense arrays

Brandon Benvie recently mentioned a trick for creating a dense array on the es-discuss mailing list:

```
> var a = Array.apply(null, Array(3));
> a
[ undefined, undefined, undefined ]
```

The above invocation is equivalent to

```
Array(undefined, undefined, undefined)
```

For many things, there is not much of a difference between this array and the previous sparse array:

```
> a.length
3
> a[0]
undefined
```

However, you can now iterate over the elements, e.g. to fill the array with values:

```
> a.forEach(function (x, i) { console.log(i+". "+x) });
0. undefined
1. undefined
2. undefined
```

Dr. Axel Rauschmayer

```
> a.map(function (x, i) { return i })
[ 0, 1, 2 ]
```

3. One more trick

The email also mentions the following trick:

```
> Array.apply(null, Array(3)).map(Function.prototype.call.bind(Number))
[ 0, 1, 2 ]
```

This is roughly the same as

```
Array.apply(null, Array(3)).map(
    function (x,i,...) { return Number.call(x,i,...) })
```

Note that `x` is the first parameter of `call` and specifies the value of `this`. `Number` being a function, that value is ignored. I prefer the more explicit variant shown above:

```
Array.apply(null, Array(3)).map(function (x,i) { return i })
```

4. Useful in practice?

In practice, creating a dense array in the manner described above will make your code difficult to understand for others. It is thus better to use utility functions such as `_.range`:

```
> _.range(3)
[ 0, 1, 2 ]
```

Combine it with `map`, in order to fill an array with a given value.

```
> _.range(3).map(function () { return "a" })
[ 'a', 'a', 'a' ]
```

5. Related posts

[1] Iterating over arrays and objects in JavaScript

[2] Trying out Underscore on Node.js

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

Recommend 4 **Share** **Sort by Best**

Join the discussion…

Dmitri Pavlutin • a year ago

Nice article! Short and clean :).

Array.prototype.filter function skips non exist elements, thus it can be used to transform sparse to dense arrays.
For example:

```
var sparse = [0, , ,];
var dense = sparse.filter(function() {
  return true;
});
```

2 ^ | v • Reply • Share ›

Šime Vidas • 4 years ago

I've found that other values also work. E.g.

Array.apply(0, Array(3))

Since, 0 is only one character, maybe use it instead of null? Or is there a particular reason why null is used?

^ | v • Reply • Share ›

Cory Gross • 4 years ago

Hi there, nice article! I just wanted to let you know that you have a minor

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

error at the very beginning. Believe it or not, `new Array(3);` will actually produce an array equivalent to `[ , , , ]`, rather than an array equivalent to `[ , , ]`. Also in your second code block, `a.map` would also return an array that looks like this `[ , , , ]` as well. Take the following code:

```
var a = [ , , ];
console.log(a.length);  // prints 2
var a = new Array(3);
console.log(a.length); // prints 3
var a = [ , , , ];
console.log(a.length); // prints 3
```

This is not what you would expect, but it leads from the following:

```
var a = [ ];
console.log(a.length);  // prints 0
a = [ , ];
console.log(a.length); // prints 1
```

Basically there is an implied `undefined` before each comma you add, but not there is not an implied `undefined` after the last comma. I didn't know this myself until I messed with this in the console just now.

^ | v • Reply • Share ›

KHS ➦ Cory Gross • 3 years ago

It's because, as you correctly noted, the value after the last comma is not an implied `undefined`. Try this:

```
a = [0, 1, 2, ]
console.log(a.length);  // 3
console.log(a);         // [0, 1, 2]
```

^ | v • Reply • Share ›

Axel Rauschmayer Mod ➦ Cory Gross • 4 years ago

Indeed! But that’s an issue with how the Node.js REPL prints arrays. Everything that is typed in is correct. Example interaction in the REPL:

```
> [ , ,, ]
[ , ,  ]
```

Maybe I should manually “fix” the output, I’ll think about it.

^ | v • Reply • Share ›

Cory Gross ➦ Axel Rauschmayer • 4 years ago

Ah I see, I was experimenting in Chrome's console. In Chrome it returns:

```
> [ , , , ]
[undefined x 3]
```

^ | v • Reply • Share ›

✉ Subscribe   Add Disqus to your site Add Disqus Add   🔒 Privacy

Newer Post   Home   Older Post

▶ January (19)

▶ 2011 (380)

▶ 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)
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)

Subscribe to: Post Comments (Atom)

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)