②ality – JavaScript and more

Free email newsletter: “ES.next News”

2013-11-23

# Initializing an array with values

Labels: dev, javascript, jsarrays, jslang

It is not a frequent use case, but it comes up occasionally: Producing an array [1] of a given length that is filled with values. This blog post explains how to do it and what to watch out for.

Let us start with something simple: producing an array of length *n* whose first element is 0, second element is 1, etc.

## 1. Array.prototype.map()

Once you have an array with length *n*, you can use the array method `map()` to fill it appropriately. For example, to produce the array `[0, 1, 2]`, any array of length 3 will do:

```
> var arr = [null, null, null];
> arr.map(function (x, i) { return i })
[ 0, 1, 2 ]
```

Alas, `map()` skips holes, but preserves them [2], which is why an empty array (with only holes) does not work for us:

```
> new Array(3).map(function (x, i) { return i })
[ , ,  ]
```

The next two sections look at ways of creating an array with a given length.

## 2. Filling an array via apply()

`Function.prototype.apply()` treats holes as if they were `undefined` elements [3]. Therefore, the following two expressions are equivalent.

```
Array(undefined, undefined, undefined)
Array.apply(null, Array(3))
```

If we combine this trick with `map()`, we get what we wanted:

```
function fillArrayWithNumbers(n) {
    var arr = Array.apply(null, Array(n));
    return arr.map(function (x, i) { return i });
}
```

The function in action:

```
> fillArrayWithNumbers(5)
[ 0, 1, 2, 3, 4 ]
```

## 3. _.range()

Underscore.js is the most popular way of complementing JavaScript’s Spartan standard library. It comes with the function `range()` which does what we want:

```
> _.range(5)
[ 0, 1, 2, 3, 4 ]
```

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)

Dr. Axel Rauschmayer

Free online books by Axel

Speaking JavaScript [up to ES5]

Exploring ES6

JavaScript training: Ecmanauten

- ► 2015 (65)
- ► 2014 (55)
- ▼ 2013 (98)
  - ► December (6)
  - ▼ November (3)
    - Initializing an array with values
    - ECMAScript 6 modules in future browsers
    - Immediately invoked constructors and object litera...
  - ► October (4)
  - ► September (6)
  - ► August (6)
  - ► July (8)
  - ► June (11)
  - ► May (11)
  - ► April (12)
  - ► March (8)
  - ► February (10)
  - ► January (13)
- ► 2012 (177)
- ► 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)

The previous trick with `apply()` is not very self-explanatory and Underscore.js is as close to a standard as it gets. Therefore, it is best to create filled arrays via `_.range()`. If we want to fill the array with something other than a range of numbers, we can use `map()`:

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

## 4. Setting up multi-dimensional arrays

If you need multiple dimensions for elements, you must nest arrays. When you create such nested arrays, the innermost arrays can grow as needed. But if you want direct access to all elements, you need to create the outer arrays.

Let's use `_.range()` to set up a matrix for Tic-tac-toe. We initialize all arrays correctly:

```
var ticTacToe = _.range(3).map(function () {
    // Create one row
    return _.range(3).map(function () {
        return '.';
    });
});
ticTacToe[0][2] = 'X';  // [row][column]
ticTacToe.forEach(function (row) {
    console.log(row.join(' '));
});
```

Output:

```
. . X
. . .
. . .
```

## 5. References

[1] Arrays in JavaScript

[2] Array iteration and holes in JavaScript

[3] Apply and arrays: three tricks

15 Comments The 2ality blog 1 Login

Recommend 5 Share Sort by Best

Join the discussion…

**Guest** • 3 years ago

You can also use _.times(n, function(n) { ... } ) instead of _.range(n).map(function(){ ... }

I find it really useful when testing applications to quickly and clearly build mock-up arrays.

3 ^ | v • Reply • Share ›

**PaulGrenier** • 2 years ago

I think ES6 gives us another option:

console.log([...Array(3)].map((x, i) => i))

2 ^ | v • Reply • Share ›

**Caleb Boyd** ➦ PaulGrenier • a year ago

Looks like we also have:

Array.from(Array(3), (x, i) => i)

4 ^ | v • Reply • Share ›

**Victor Homyakov** • 2 years ago

Tweets by @rauschma

**Axel Rauschmayer** @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…

13h

**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

- 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)

**Victor Homyakov** • 2 years ago

ES6 proposes Array(desiredLength).fill(value)
http://people.mozilla.org/~jor...

2 ^ | v • Reply • Share ›

**stri8ed** ➦ Victor Homyakov • a year ago

This does not work as expected for Objects. E.g. Array(3).fill([]); will store the same array in each index.

^ | v • Reply • Share ›

**Victor Homyakov** ➦ stri8ed • a year ago

Why not as expected? Any other implementation wrapped in a function will do the same, e.g.
function fill(arr, value) { return arr.map(() => value) }
just because objects are copied by reference

^ | v • Reply • Share ›

**Renzo Kooi** • 2 years ago

Array.apply(null, Array(n)) with large n will exceed the maximum call stack. In that case Array(max).join().split(',') would give a 'mappable' Array. That has a limit too: it will eventually reach the (browser) boundary of too large a string (from the join operation). See http://jsfiddle.net/KooiInc/v3...

1 ^ | v • Reply • Share ›

**Mahir Shah** • 2 years ago

We could shorten if we are allowed to use Object.keys:

Object.keys(Array.apply(null, Array(10)))

1 ^ | v • Reply • Share ›

**Axel Rauschmayer** Mod ➦ Mahir Shah • 2 years ago

Clever. But note that the elements are all strings (which may or may not be what you want).

^ | v • Reply • Share ›

**Mahir Shah** ➦ Axel Rauschmayer • 2 years ago

You are correct! Updated:
Object.keys(new Int8Array(10)).map(parseFloat)

^ | v • Reply • Share ›

**obedm503** • a month ago

one could also use:

```
> Array(3).fill('a')
< ["a", "a", "a"]
```

I find it's shorter and easier to understand if you know that the Array constructor creates an array of a given length if only one integer is passed.

^ | v • Reply • Share ›

**whycantibeanon** • 2 years ago

Array.fill should be available in the modern browsers today:

var myArray = new Array( 10 ).fill( 'blah' );

^ | v • Reply • Share ›

**Semmel** • 3 years ago

I admit _.range() is nice especially for anonymous arrays.

For the other use cases I would prefer simple for loops, imho they are simpler and more readable

```
for (i = 0; i < 10; i++){
arr[i] = []
for (j = 0; j < 3; j++){
```

```
arr[i][j] = i*j;
}
}
```

^ | v • Reply • Share ›

**Guest** • 3 years ago

The callback to Array.map is invoked with 3 arguments, when would you ever use the 3rd argument–the Array object being traversed?

^ | v • Reply • Share ›

**Axel Rauschmayer** Mod ➦ Guest • 3 years ago

If you want the callback to be reusable for several arrays and it needs to access elements other than the current one.

3 ^ | v • Reply • Share ›

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

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

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)