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)

2015-09-13

ECMAScript 6: holes in Arrays

Labels: dev, esnext, javascript

This blog post describes how ECMAScript 6 handles holes in Arrays.

1. Holes in Arrays

Holes are indices "inside" an Array that have no associated element. In other words: An Array `arr` is said to have a hole at index `i` if:

- `0 ≤ i < arr.length`
- `!(i in arr)`

For example: The following Array has a hole at index 1.

```
> let arr = ['a',,'b']
'use strict'
> 0 in arr
true
> 1 in arr
false
> 2 in arr
true
> arr[1]
undefined
```

For more information, consult Sect. "Holes in Arrays" in "Speaking JavaScript".

2. ECMAScript 6: holes are treated like `undefined` elements

The general rule for Array methods that are new in ES6 is: each hole is treated as if it were the element `undefined`. Examples:

```
> Array.from(['a',,'b'])
[ 'a', undefined, 'b' ]
> [,'a'].findIndex(x => true)
0
> [...[,'a'].entries()]
[ [ 0, undefined ], [ 1, 'a' ] ]
```

The idea is to steer people away from holes and to simplify long-term. Unfortunately that means that things are even more inconsistent now.

3. Array methods and holes

3.1. `Array.from()`

`Array.from()` converts holes to `undefined`:

```
> Array.from(['a',,'b'])
[ 'a', undefined, 'b' ]
```

With a second argument, it works mostly like `map()`, but does not ignore holes:

Dr. Axel Rauschmayer

Free online books by Axel

Speaking JavaScript [up to ES5]

Exploring ES6

JavaScript training:
Ecmanauten

▼ 2015 (65)
► December (7)
► November (6)
► October (13)
▼ September (5)
Customizing ES6 via well-known symbols
__proto__ in ECMAScript 6
ECMAScript 6: holes in Arrays
The names of functions in ES6
Typed Arrays in ECMAScript 6
► August (6)
► July (3)
► June (3)
► April (4)
► March (4)
► February (8)
► January (6)
► 2014 (55)
► 2013 (98)
► 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)

```
> Array.from(new Array(3), (x,i) => i)
[ 0, 1, 2 ]
```

3.2. Spread operator (`...`)

Inside Arrays, the spread operator (`...`) works much like `Array.from()` (but its operand must be iterable, whereas `Array.from()` can handle anything that's Array-like).

```
> [...['a',,'b']]
[ 'a', undefined, 'b' ]
```

3.3. `Array.prototype` methods

In ECMAScript 5, behavior already varied slightly. For example:

- `forEach()`, `filter()`, `every()` and `some()` ignore holes.
- `map()` skips but preserves holes.
- `join()` and `toString()` treat holes as if they were `undefined` elements, but interprets both `null` and `undefined` as empty strings.

ECMAScript 6 adds new kinds of behaviors:

- `copyWithin()` creates holes when copying holes (i.e., it deletes elements if necessary).
- `entries()`, `keys()`, `values()` treat each hole as if it was the element `undefined`.
- `find()` and `findIndex()` do the same.
- `fill()` doesn't care whether there are elements at indices or not.

The following table describes how `Array.prototype` methods handle holes.

Method	Holes are	
concat	Preserved	['a',,'b'].concat(['c',,'d']) → ['a',,'b','c',,'d']
copyWithin ✓	Preserved	[,'a','b',,].copyWithin(2,0) → [,'a',,'a']
entries ✓	Elements	[...[,'a'].entries()] → [[0,undefined], [1,'a']]
every	Ignored	[,'a'].every(x => x==='a') → true
fill ✓	Filled	new Array(3).fill('a') → ['a','a','a']
filter	Removed	['a',,'b'].filter(x => true) → ['a','b']
find ✓	Elements	[,'a'].find(x => true) → undefined
findIndex ✓	Elements	[,'a'].findIndex(x => true) → 0
forEach	Ignored	[,'a'].forEach((x,i) => log(i)); → 1
indexOf	Ignored	[,'a'].indexOf(undefined) → -1
join	Elements	[,'a',undefined,null].join('#') → '#a##'
keys ✓	Elements	[...[,'a'].keys()] → [0,1]
lastIndexOf	Ignored	[,'a'].lastIndexOf(undefined) → -1
map	Preserved	[,'a'].map(x => 1) → [,1]
pop	Elements	['a',,].pop() → undefined
push	Preserved	new Array(1).push('a') → 2

Method	Holes are	
`reduce`	Ignored	`['#',,undefined].reduce((x,y)=>x+y) → '#undefined'`
`reduceRight`	Ignored	`['#',,undefined].reduceRight((x,y)=>x+y) → 'undefined#'`
`reverse`	Preserved	`['a',,'b'].reverse() → ['b',,'a']`
`shift`	Elements	`[,'a'].shift() → undefined`
`slice`	Preserved	`[,'a'].slice(0,1) → [,]`
`some`	Ignored	`[,'a'].some(x => x !== 'a') → false`
`sort`	Preserved	`[,undefined,'a'].sort() → ['a',undefined,,]`
`splice`	Preserved	`['a',,].splice(1,1) → [,]`
`toString`	Elements	`[,'a',undefined,null].toString() → ',a,,'`
`unshift`	Preserved	`[,'a'].unshift('b') → 3`
`values` ✓	Elements	`[...[,'a'].values()] → [undefined,'a']`

Notes:

- ES6 methods have checkmarks (✓).
- JavaScript ignores a trailing comma in an Array literal: `['a',,].length → 2`
- Helper function used in the table: `const log = console.log.bind(console);`

4. Recommendations

With regard to holes in Arrays, the only rule is now that there are no rules. Therefore, you should avoid holes if you can (they affect performance negatively, too). If you can't then the table in the previous section may help.

5. Further reading

- ECMAScript 5: Chapter "Arrays" in "Speaking JavaScript"
- ECMAScript 6: Chapter "New Array features" in "Exploring ES6"

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)

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

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)