# ②ality – JavaScript and more

About | Donate | Subscribe | ES2017 | Books (free online!)

## 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)

Free email newsletter: "ES.next News"

2016-02-24

# Arrow functions vs. bind()

Labels: coding, dev, esnext, javascript

ES6 arrow functions are often a compelling alternative to `Function.prototype.bind()`.

## 1. Extracting methods

If an extracted method is to work as a callback, you must specify a fixed `this`, otherwise it will be invoked as a function (and `this` will be `undefined` or the global object). For example:

```
obj.on('anEvent', console.log.bind(console))
```

An alternative is to use an arrow function:

```
obj.on('anEvent', x => console.log(x))
```

## 2. `this` via parameters

The following code demonstrates a neat trick: For some methods, you don't need `bind()` for a callback, because they let you specify the value of `this`, via an additional parameter. `filter()` is one such method:

```
const as = new Set([1, 2, 3]);
const bs = new Set([3, 2, 4]);
const intersection = [...as].filter(bs.has, bs);
    // [2, 3]
```

However, this code is easier to understand if you use an arrow function:

```
const as = new Set([1, 2, 3]);
const bs = new Set([3, 2, 4]);
const intersection = [...as].filter(a => bs.has(a));
    // [2, 3]
```

## 3. Partial evaluation

`bind()` enables you to do *partial evaluation*, you can create new functions by filling in parameters of an existing function:

```
function add(x, y) {
    return x + y;
}
const plus1 = add.bind(undefined, 1);
```

Again, I find an arrow function easier to understand:

```
const plus1 = y => add(1, y);
```

## 4. Further reading

- Currying versus partial application (with JavaScript code)

Dr. Axel Rauschmayer

## Free online books by Axel

- Speaking JavaScript [up to ES5]
- Exploring ES6

**JavaScript training:** Ecmanauten

- ▶ December (1)
- ▶ November (4)
- ▶ October (4)
- ▶ September (5)
- ▶ August (1)
- ▶ June (1)
- ▶ May (2)
- ▶ April (2)
- ▶ March (2)
- ▼ February (8)
  - Arrow functions vs. bind()
  - Examples of name clashes in JavaScript's standard …
  - JavaScript fatigue fatigue
  - ES proposal: Object.getOwnPropertyDescriptors()
  - ECMAScript 2017: the final feature set
  - ES proposal: async functions
  - ES2016 feature: exponentiation operator (**)
  - ES2016 feature: Array.prototype.includes
- ▶ January (8)

- ▶ 2015 (65)
- ▶ 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)

- Uncurrying `this` in JavaScript

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

**Recommend** 6 **Share** **Sort by Best**

Join the discussion…

**bradleymeck** • a year ago

Don't forget `arguments` is not defined in an arrow function as well.

3 ^ | ⌄ • Reply • Share ›

**Axel Rauschmayer** Mod ➦ bradleymeck • a year ago

Right. But that's a different topic. Rest parameters should (mostly) make up for that, though.

6 ^ | ⌄ • Reply • Share ›

**zenw0lf** • a year ago

A minor nitpick: what you refer as partial evaluation is in reality partial (function) application. https://en.wikipedia.org/wiki/...

2 ^ | ⌄ • Reply • Share ›

**Barney Carroll** • a year ago

Regarding partial evaluation: a bound function can still have `this` re-bound, which is rarely desirable. Arrows ensure a total lockdown of `this`.

1 ^ | ⌄ • Reply • Share ›

**Nikos** • 8 months ago

I've had zero luck arrow functions and babel recently, having to resort to .bind(this) all over the place

^ | ⌄ • Reply • Share ›

**dou4cc** • a year ago

I think it is better below:
::console.log

^ | ⌄ • Reply • Share ›

**virtyaluk** ➦ dou4cc • 8 months ago

console.log is now bound to console object. So there's no need to use that syntax or even arrow function syntax. Actually, you may write something like this
obj.on('anEvent', console.log)
and it's just a code optimization.

Many of those who write this kind of articles demonstrate non-optimized code with the only purpose to show the new syntax. That's stupid. You may write elegant and optimized code avoiding this kind of useless advice ("Extracting methods" section) most of the time.

^ | ⌄ • Reply • Share ›

**Axel Rauschmayer** Mod ➦ virtyaluk • 8 months ago

This blog post is not about performance or about showing new syntax, but about writing code that's as easy to understand as possible. Additionally, engines are getting better and better at optimizing these kinds of patterns, so I wouldn't even be sure about performance in the long run.

^ | ⌄ • Reply • Share ›

**Drew | Person** ➦ dou4cc • a year ago

I'm excited for this syntax too. I wonder how quickly it will progress.

## Tweets by @rauschma

**Axel Rauschmayer** @rauschma

Enjoying "Troll Hunters":
– "Let's call him 'Gnome Chomsky'"
– "Juliet dies in this? Nooo!"

9h

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

- 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)

^ | v • Reply • Share ›

**Anas Raza Firdousi** ➦ Drew | Person • a year ago

Already available in things like TypeScript

^ | v • Reply • Share ›

**Šime Vidas** • a year ago

There’s the code repetition, though. It seems ok for when there’s only one parameter, but when you have 3 or more, it becomes tedious (and also error-prone).

^ | v • Reply • Share ›

**Dmitri Pavlutin** • a year ago

Well, when using bind() you don't create a closure of the current scope. But I found arrow functions pretty too :).

^ | v • Reply • Share ›

**olexandr** ➦ Dmitri Pavlutin • 10 months ago

Using bind() will lead to performance degradation because of switching between JIT and c++ code at least for JSC. I think the same for V8 and Chakra and SpiderMonkey. Arrow function will be executed without such switches.
Also as I know .call() and .apply() are faster to .bind()

^ | v • Reply • Share ›

**Axel Rauschmayer** Mod ➦ Dmitri Pavlutin • a year ago

There is much that engines could optimize, so we’ll see what happens in the long run.

^ | v • Reply • Share ›

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

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

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)