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)

2011-04-20

Iterating over arrays and objects in JavaScript

Labels: dev, javascript, jslang

This post explains three approaches for extracting information from arrays and objects:

1. for loops,
2. array methods (courtesy of ECMAScript 5 [1]),
3. listing property keys.

It concludes with best practices for applying these approaches.

1. for loops

All for loops can be used with the following statements.

- break [label]: exit from a loop.
- continue [label]: stop the current loop iteration, immediately continue with the next one.
- label: A label is an identifier followed by a colon. In front of a loop, a label allows you to break or continue that loop even from a loop nested inside of it. In front of a block, you can break out of that block. In both cases the name of the label becomes an argument of `break` or `continue`. Example for breaking out of a block:

```
function findEvenNumber(arr) {
    loop: { // label
        for(var i=0; i<arr.length; i++) {
            if ((arr[i] % 2) === 0) {
                console.log("Found: "+arr[i]);
                break loop;
            }
        }
        console.log("No even number found.");
    }
    console.log("DONE");
}
```

1.1. for

Syntax:

```
for ([start]; [condition]; [final-expression])
    statement
```

Rules:

- Traditional way of iterating over arrays.
- Can use `var`, but scope is always the complete surrounding function.

Example:

```
var arr = [ "a", "b", "c" ];
for(var i=0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

Dr. Axel Rauschmayer

1.2. for...in

Syntax

```
for (variable in object)
    statement
```

Rules:

- Iterate over property keys, including inherited ones.
- Don’t use for arrays. It iterates over both array indices and property keys. There will thus be problems as soon as someone adds a property to an array.
- Can use `var`, but scope is always the complete surrounding function.
- Properties can be deleted during iteration.

Pitfall: Iterates over both array indices and property keys.

```
> var arr = [ "a", "b", "c" ];
> arr.foo = true;
> for(var key in arr) { console.log(key); }
0
1
2
foo
```

Pitfall: Iterates over inherited properties.

```
function Person(name) {
    this.name = name;
}
Person.prototype = {
    describe: function() {
        return "Name: "+this.name;
    }
};
var person = new Person("Jane");
for(var key in person) {
    console.log(key);
}
```

Output:

```
name
describe
```

Skip inherited properties: via `hasOwnProperty()`.

```
for(var key in person) {
    if (person.hasOwnProperty(key)) {
        console.log(key);
    }
}
```

1.3. for each...in

Non-standard (Firefox only), iterates over the values of an object. Don’t use it.

2. Array methods for iteration

2.1. Iterate

Iterate over the elements in an array. The methods don’t have a result, but you can produce one in the callback as a side effect. They all have the following signature:

```
function(callback, [thisValue])
```

Parameters:

- The callback has the following signature (sometimes it returns no value, sometimes a boolean).

  ```
  function([element], [index], [collection])
  ```

- The `thisValue` argument allows you to specify an object that is to be accessed via `this` in `callback`.

Iteration methods:

▶ 2015 (65)

▶ 2014 (55)

▶ 2013 (98)

▶ 2012 (177)

▼ 2011 (380)

▶ December (22)

▶ November (25)

▶ October (27)

▶ September (15)

▶ August (24)

▶ July (18)

▶ June (29)

▶ May (23)

▼ April (33)

Creating new programming languages is becoming eas...

ECMAScript.next features are taking shape

Patterns for modules and namespaces in JavaScript

George R. R. Martin: a successful fantasy author a...

The Sony S2 tablet might make dual screens work, t...

New Honeycomb tablet from Lenovo this summer, with...

ECMAScript 5 spec: LexicalEnvironment versus Varia...

Motorola is working on its own mobile OS

iPhones and iPads are keeping a log of where you h...

Iterating over arrays and objects in JavaScript

Song: “From where I’m standing” by Schuyler Fisk

The truth behind the seemingly too honest URL of a...

Are Amazon and Apple playing chicken regarding in-...

JavaScript: converting any

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

- `Array.prototype.forEach()` is similar to `for...in`, but only iterates over an object's own properties.
- `Array.prototype.every()`: returns `true` if the callback returns `true` for every element.
- `Array.prototype.some()`: returns `true` if the callback returns true for at least one element.

Example:

```
var arr = [ "apple", "pear", "orange" ];
arr.forEach(function(elem) {
    console.log(elem);
});
```

Pitfall: `forEach()` does not support `break`. Use `every()` instead:

```
function breakAtEmptyString(arr) {
    arr.every(function(elem) {
        if (elem.length === 0) {
            return false; // break
        }
        console.log(elem);
        return true; // don't forget!
    });
}
```

`every()` returns `false` if a break happened and `true`, otherwise. This allows you to react to the iteration finishing successfully (something that is slightly tricky with `for` loops). Caveat: You need to return a "true" value to keep going. If you want to avoid that, you can use `some()` and return `true` to break:

```
function breakAtEmptyString(arr) {
    arr.some(function(elem) {
        if (elem.length === 0) {
            return true; // break
        }
        console.log(elem);
        // implicit: return undefined (interpreted as false)
    });
}
```

2.2. Transform

Transformation methods take an input array and produce an output array, while the callback controls how the output is produced. The callback has the same signature as for iteration:

```
function([element], [index], [collection])
```

Methods:

- `Array.prototype.map(callback, [thisValue])`: Each output array element is the result of applying `callback` to an input element.
- `Array.prototype.filter(callback, [thisValue])`: The output array contains only those input elements for which `callback` returns true.

2.3. Reduce

For reducing, the callback has a different signature:

```
function(previousElement, currentElement, currentIndex, collection)
```

Methods:

- `Array.prototype.reduce(callback, [initialValue])`: Compute a value by applying `callback` to pairs (`previousElement`, `currentElement`) of array elements.
- `Array.prototype.reduceRight(callback, [initialValue])`: Same as `reduce()`, but from right to left.

Example:

```
// Sum of all array elements:
[17, 5, 4, 28].reduce(function(prev, cur) {
    return prev + cur;
});
```

► 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)
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)

3. Listing property keys

- `Object.keys(obj)`: Lists all enumerable own property keys of an object. Example:

```
> Object.keys({ first: "John", last: "Doe" })
[ 'first', 'last' ]
```

- `Object.getOwnPropertyNames()`: Lists all own property keys of an object, including non-enumerable ones.

```
> Object.getOwnPropertyNames(Number.prototype)
[ 'toExponential'
, 'toString'
, 'toLocaleString'
, 'toPrecision'
, 'valueOf'
, 'toJSON'
, 'constructor'
, 'toFixed'
]
> Object.keys(Number.prototype)
[]
```

  Comment: The main reason that prototype methods are not enumerable is to hide them from iteration mechanisms that include inherited properties.

4. Best practices

Iterating over arrays

Options:

- Simple `for` loop.
- One of the iteration methods.
- Never use `for...in` or `foreach...in`.

Iterating over objects

Options:

- Combine `for...in` with `hasOwnProperty()`, in the manner described above.
- Combine `Object.keys()` or `Object.getOwnPropertyNames()` with `forEach()` array iteration.

```
var obj = { first: "John", last: "Doe" };
// Visit non-inherited enumerable keys
Object.keys(obj).forEach(function(key) {
    console.log(key);
});
```

Other tasks:

- Iterate over the property (key,value) pairs of an object: Iterate over the keys, use each key to retrieve the corresponding value. Other languages make this simpler, but not JavaScript.

5. Related reading

1. What's new in ECMAScript 5

12 Comments The 2ality blog 1 Login

Recommend 6 Share Sort by Best

Join the discussion…

PAEz • 3 years ago
http://jsperf.com/iterating-ov...
1 ^ | v • Reply • Share ›

algore87 • a year ago
how to iterate over indices with intention to remove items in an array while iterating over them?

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)

• Reply • Share ›

Jesse Silva • 2 years ago

Thanks a lot.

In this last example: Iterating over objects

How can i iterate with child:
var obj = {
first: "John",
last: "Doe",
child: [
{
first: "Jane",
last: "Doe"
}
]
};

• Reply • Share ›

Axel Rauschmayer Mod ➦ Jesse Silva • 2 years ago

Nested loops?

• Reply • Share ›

Jesse Silva ➦ Axel Rauschmayer • 2 years ago

I guess.. i'm trying to return child objects and i can't figure this out. In your example above, how i return in the console.log the child?

Thanks

• Reply • Share ›

bryc • 2 years ago

`forEach` is slower than iterating over `Object.keys(..)` using a standard `for loop`. Only barely though.

`for-in and hasOwnProperty()` turns out to be significantly slower, which is surprising.

http://jsperf.com/for-in-versu...

• Reply • Share ›

Khoguan • 3 years ago

Thanks a lot.
I found a small bug:

example for 1.1

var arr = ["a", "b", "c"];
for(var i=0; i<0; i++) {
console.log(arr[i]);
}

The condition 'i<0' is wrong.

• Reply • Share ›

Axel Rauschmayer Mod ➦ Khoguan • 2 years ago

Fixed, thanks!

• Reply • Share ›

Michael ➦ Khoguan • 2 years ago

Curious about that also; thought maybe it was a trick I didn't know about.

• Reply • Share ›

Axel Rauschmayer Mod ➦ Michael • 2 years ago

Axel Rauschmayer Mod Michael • 2 years ago

No, just a boring old typo. ;-)

Reply • Share ›

Boldewyn • 5 years ago

Thanks for this great overview to an annoying problem!

Short correction: In the last example in 2.1 the method sould be some(), not every() (or otherwise the function name doesn't make sense).

Reply • Share ›

Axel Rauschmayer Mod Boldewyn • 4 years ago

Fixed (a while ago). Thanks!

Reply • Share ›

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

Newer Post Home Older Post

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)