Generator comprehensions

by 5 contributors:     

> **This is an experimental technology, part of the ECMAScript 2016 (ES7) proposal.**
> Because this technology's specification has not stabilized, check the compatibility table for usage in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future version of browsers as the spec changes.

The **generator comprehension** syntax is a JavaScript expression which allows you to quickly assemble a new generator function based on an existing iterable object. Comprehensions exist in many programming languages and the upcoming ECMAScript 7 standard defines array comprehensions for JavaScript.

See below for differences to the old generator expression syntax in SpiderMonkey, based on proposals for ECMAScript 4.

Syntax

```
(for (x of iterable) x)
(for (x of iterable) if (condition) x)
(for (x of iterable) for (y of iterable) x + y)
```

Description

Inside generator comprehensions, these two kinds of components are allowed:

- `for...of` and

- `if`

The `for-of` iteration is always the first component. Multiple `for-of` iterations or if statements are allowed.

Examples

Simple generator comprehensions

```
(for (i of [ 1, 2, 3 ]) i*i );
// generator function which yields 1, 4, and 9

[...(for (i of [ 1, 2, 3 ]) i*i )];
// [1, 4, 9]

var abc = [ "A", "B", "C" ];
(for (letters of abc) letters.toLowerCase());
// generator function which yields "a", "b", and "c"
```

Generator comprehensions with if statement

```
var years = [ 1954, 1974, 1990, 2006, 2010, 2014 ];

(for (year of years) if (year > 2000) year);
// generator function which yields 2006, 2010, and 2014

(for (year of years) if (year > 2000) if(year < 2010) year);
// generator function which yields 2006, the same as below:

(for (year of years) if (year > 2000 && year < 2010) year);
// generator function which yields 2006
```

Generator comprehensions compared to generator function

An easy way to understand generator comprehension syntax, is to compare it with the generator function.

Example 1: Simple generator.

```
var numbers = [ 1, 2, 3 ];

// Generator function
(function*() {
  for (let i of numbers) {
    yield i * i;
  }
})()

// Generator comprehension
(for (i of numbers) i*i );

// Result: both return a generator which yields [ 1, 4, 9 ]
```

Example 2: Using `if` in generator.

```
var numbers = [ 1, 2, 3 ];

// Generator function
(function*() {
  for (let i of numbers) {
    if (i < 3) {
      yield i * 1;
    }
  }
})()

// Generator comprehension
(for (i of numbers) if (i < 3) i);

// Result: both return a generator which yields [ 1, 2 ]
```

Specifications

Generator comprehensions were initially in the ECMAScript 6 draft, but got removed in revision 27 (August 2014). Please see older revisions of ES6 for specification semantics. An updated version is expected to be back in a new ES2016 / ES7 draft.

Browser compatibility

Desktop Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	Not supported	30 (30)	Not supported	Not supported	Not supported

SpiderMonkey-specific implementation notes

- `let` as an identifier is not supported as `let` is currently only available to JS version 1.7 and XUL scripts tags.
- Destructuring in comprehensions is not supported yet (bug 980828).

Differences to the older JS1.7/JS1.8 comprehensions

- ES7 comprehensions create one scope per "for" node instead of the comprehension as a whole.
  - Old: `[...(()=>x for (x of [0, 1, 2]))][1]() // 2`
  - New: `[...(for (x of [0, 1, 2]) ()=>x)][1]() // 1, each iteration creates a fresh binding for x.`
- ES7 comprehensions start with "for" instead of the assignment expression.
  - Old: `(i * 2 for (i of numbers))`
  - New: `(for (i of numbers) i * 2)`
- ES7 comprehensions can have multiple `if` and `for` components.
- ES7 comprehensions only work with `for...of` and not with `for...in` iterations.

See also

- for...of
- Array comprehensions