node JS® Interactive

EUROPE

Amsterdam, Netherlands
September 15–18, 2016

ES6 is so 2015, meet ES2016!

Paul Verbeek, Booking.com

Booking.com

workingatbooking.com

nlhtml5.org

ECMAScript

"ECMAScript was always an unwanted trade name that sounds like a skin disease.

Brendan Eich, creator of JavaScript

https://mail.mozilla.org/pipermail/es-discuss/2006-October/000133.html

The TC39 process

The TC39 process

Stage 0: Strawman
Free-form ideas, reviewed in TC39 meetings

Stage 1: Proposal
Formally accepted proposal

Stage 2: Draft
Has description of syntax and semantics

Stage 3: Candidate
Spec text complete, has at least 2 implementations

Stage 4: Finished
Ready for standard, passes unit tests

ECMAScript 2016

ECMA-262 7th edition

The ECMAScript® 2016 Language Specification

Exponentiation Operator

```
x ** y
Math.pow(x, y);

let squared = 2 ** 2;
// same as  2 * 2

let cubed = 3;
cubed **= 3;
// same as  cubed = 3 * 3 * 3
```

Exponentiation Operator

Array.prototype.includes

```
['a', 'b', 'c'].indexOf('a') >= 0; // true
['a', 'b', 'c'].indexOf('d') >= 0; // false
```

```
['a', 'b', 'c'].includes('a'); // true
['a', 'b', 'c'].includes('d'); // false
```

```
['a', 'b', NaN].indexOf(NaN) >= 0; // false
[‘a', 'b', NaN].includes(NaN); // true
```

```
[ , , ].indexOf(undefined) >= 0; // false
[ , , ].includes(undefined); // true
```

Wait? `inclu` **Why not**

`tains?`

Array.prototype.includes





ECMA-262 8th edition

The ECMAScript® 2017 Language Specification

Not really…

Stage 4: Finished

Ready for standard, passes unit tests

- Object.values/Object.entries
- String padding
- Object.getOwnPropertyDescriptors()
- Async functions
- Trailing commas in function parameter lists and calls

https://github.com/tc39/proposals/blob/master/finished-proposals.md

Object.values / Object.entries

```
const obj = { name: 'Paul', age: 30 };

let keys = Object.keys(obj);
// ['name', 'age']

let values = Object.values(obj);
// ['Paul', 30]

let entries = Object.entries(obj);
// [['name', 'Paul'], ['age', 30]]

for (let [key,value] of Object.entries(obj)) {
    console.log(`${key}: ${value}`);
}

// 'name': 'Paul'
// 'age': 30
```

Object.values / Object.entries

String padding

```
'1'.padStart(3, '0');
// '001'

'1'.padEnd(3, '0');
// '100'
```

```
'Node'.padEnd(9, 'JS');
// 'NodeJSJSJ'


'foo'.padStart(8);
// '     foo'
```

String padding

Object.getOwnPropertyDescriptors()

```
const obj = {
    [Symbol('foo')]: 123,
    get bar() { return 'abc' },
};
console.log(Object.getOwnPropertyDescriptors(obj));

// Output:
// { [Symbol('foo')]:
//    { value: 123,
//      writable: true,
//      enumerable: true,
//      configurable: true },
//   bar:
//    { get: [Function: bar],
//      set: undefined,
//      enumerable: true,
//      configurable: true } }
```

Copying properties into an object

```
var o1 = { a: 1 };
var o2 = { b: 2 };

Object.assign(o1, o2);
console.log(o1);  // { a: 1, b: 2 }
```

Copying properties into an object

```
const source = {
    set foo(value) {
        console.log(value);
    }
};
console.log(Object.getOwnPropertyDescriptor(source, 'foo'));
// { get: undefined,
//   set: [Function: foo],
//   enumerable: true,
//   configurable: true }

const target = {};
Object.assign(target, source);
console.log(Object.getOwnPropertyDescriptor(target, 'foo'));
// { value: undefined,
//   writable: true,
//   enumerable: true,
//   configurable: true }
```

Copying properties into an object

```
const source = {
    set foo(value) {
        console.log(value);
    }
};

const target = {};
Object.defineProperties(target,
Object.getOwnPropertyDescriptors(source));

console.log(Object.getOwnPropertyDescriptor(target, 'foo'));
// { get: undefined,
//   set: [Function: foo],
//   enumerable: true,
//   configurable: true }
```

Cloning objects

```
const clone = Object.create(
    Object.getPrototypeOf(obj),
    Object.getOwnPropertyDescriptors(obj)
);
```

Object.getOwnPropertyDescriptors()

Async functions

```
function doSomethingAsync(callback) {
    setTimeout(function () {
        if (typeof callback === 'function') {
            callback('something');
        }
    }, 1000)
}
```

```
function doSomethingAsync(callback) {
    setTimeout(function () {
        if (typeof callback === 'function') {
            callback('something');
        }
    }, 1000)
}

function doWork() {
    console.log('start');
    doSomethingAsync(function (value) {
        console.log(value)
    });
}

doWork();
console.log('working');
// 'start'
// 'working'
// 'something'
```

```
function doSomethingAsync() {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            resolve('something');
        }, 1000)
    });
}

function doWork() {
    console.log('start');
    doSomethingAsync().then(function (value) {
        console.log(value)
    });
}

doWork();
console.log('working');
// 'start'
// 'working'
// 'something'
```

```
function doSomethingAsync() {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            resolve('something');
        }, 1000)
    });
}

async function doWork() {
    console.log('start');
    let value = await doSomethingAsync();
    console.log(value);
}

doWork();
console.log('working');
// 'start'
// 'working'
// 'something'
```

```
function doSomethingAsync() {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            resolve('something');
        }, 1000)
    });
}

async function doWork() {
    console.log('start');
    let value = await doSomethingAsync();
    console.log(value);
    value += await doSomethingAsync();
    console.log(value);
}

doWork();
console.log('working');
// 'start'
// 'working'
// 'something'
// 'somethingsomething'
```

```
function doSomethingAsync() {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            reject('you failed! hah!');
        }, 1000)
    });
}

async function doWork() {
    console.log('start');
    let value;
    try {
        value = await doSomethingAsync();
    } catch (e) {
        value = 'error: ' + e.message;
    }
    console.log(value);
}

doWork();
console.log('working');
// 'start'
// 'working'
// 'you failed! hah!'
```

https://ponyfoo.com/articles/understanding-javascript-async-await

Async functions

Trailing commas in function parameter lists and calls

In objects and arrays

```
let obj = {
    first: 'Jane',
    last: 'Doe',
};
```

```
let arr = [
    'red',
    'green',
    'blue',
];
console.log(arr.length); // 3
```

In function parameter lists and calls

```
function foo(
    param1,
    param2,
) {}
```

```
foo(
    'abc',
    'def',
);
```

Trailing commas in function parameter lists and calls

Stage 3: Candidate

Spec text complete, has at least 2 implementations

- SIMD.JS - SIMD APIs + polyfill
- `Function.prototype.toString` revision
- Lifting Template Literal Restriction

Status: https://github.com/tc39/ecma262

Specs: https://tc39.github.io/ecma262/

Node.js ES2016 Support

kangax's compat-table applied only to Node.js

Learn more

requires harmony flag

Created by William Kapke

	Nightly! 7.0.0 100% complete	6.5.0 100% complete	5.12.0 30% complete	5.11.1 30% complete	5.11.0 30% complete	4.5.0 7% complete	4.4.7 7% complete	4.4.6 7% complete	4.3.2 7% complete	0.12.15 7% complete	0.10.46 0% complete
features											
exponentiation (**) operator											
basic support	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
assignment	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
early syntax error for unary negation without parens	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
Array.prototype.includes											
Array.prototype.includes	Yes	Yes	Yes	Yes	Yes	Error	Error	Error	Error	Error	Error
Array.prototype.includes is generic	Yes	Yes	Yes	Yes	Yes	Error	Error	Error	Error	Error	Error
%TypedArray%.prototype.includes	Yes	Yes	Yes	Yes	Yes	Error	Error	Error	Error	Error	Error
misc											
generator functions can't be used with "new"	Yes	Yes	No	No	No	No	No	No	No	No	Error
generator throw() caught by inner generator	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Error
strict fn w/ non-strict non-simple params is error	Yes	Yes	No	No	No	No	No	No	No	Error	Error
nested rest destructuring, declarations	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
nested rest destructuring, parameters	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
Proxy, "enumerate" handler removed	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
Proxy internal calls, Array.prototype.includes	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error

Node.js ES2017 Support

	Nightly! 7.0.0 41% complete	6.5.0 41% complete	5.12.0 30% complete	5.11.1 30% complete	5.11.0 30% complete	4.5.0 30% complete	4.4.7 30% complete	4.4.6 30% complete	4.3.2 30% complete	0.12.15 30% complete	0.10.46 19% complete
features											
Object.values	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
Object.entries	Yes	Yes	Error	Error	Error	Error	Error	Error	Error	Error	Error
Object.getOwnPropertyDescriptors											

http://node.green

https://ponyfoo.com/

http://www.2ality.com/

Booking.com

workingatbooking.com


Expedia®


@_paulverbeek


@nlhtml5


verbeek.p@gmail.com


4815162342

Questions?