THE PRESENT AND FUTURE OF JAVASCRIPT

2017/10/08

😋 ME 😙

YUTA HIROTO

npx hiroppy

A JS engineer and Architect
Dwango, Inc

Working on Node.js, Babel
Node.js Japan User Group

node
JS Interactive
NORTH AMERICA
October 4 - 6, 2017
Vancouver Convention Centre, West Building, Vancouver, BC Canada

Node Interactive(2017-10-04 - 2017-10-05)
and Node.js Collaborator Summit(2017-10-06 - 2017-10-07)

The events are being held now!!

contents

ECMAScript

Specification of JavaScript by Ecma International.

So... what is JavaScript?🤔

ECMAScript - JavaScript's specification(standardization)

JavaScript - implementation based on ECMAScript

TC39

Technical Committee 39

Ecma TC39

Ecma International, Technical Committee 39 - ECMAScript

http://ecma-international.org/memento/TC39.htm

Maintains ECMA-262 and ECMA-402.
Members of the committee are all major browser vendors and companies.

Editions

Edition	Name	Year
1	ECMAScript 1	06 / 1997
2	ECMAScript 2	06 / 1998
3	ECMAScript 3	12 / 1999
4	ECMAScript 4	Abandoned
5	ECMAScript 5	12 / 2009
5.1	ECMAScript 5.1	06 / 2011
6	ECMAScript 2015	06 / 2015
7	ECMAScript 2016	06 / 2016
8	ECMAScript 2017	06 / 2017

JavaScript will continue to evolve💪

Specifications are updated every year.

understand the process🤓

TC39 Process

Needs to pass five processes to formulate specifications.
It is called stage-X.(0 - 4)

TC39's meeting is held every two months and this stages will change.

Dates	Location	Host
2017-01-24 to 2017-01-26	**San Jose, CA**	**PayPal**
2017-03-21 to 2017-03-23	**Portland, OR**	**Mozilla**
2017-05-23 to 2017-05-25	**New York, NY**	**Google**
2017-07-25 to 2017-07-27	**Redmond, WA**	**Microsoft**
2017-09-26 to 2017-09-28	**Boston, MA**	**Bocoup**
2017-11-28 to 2017-11-30	**San Francisco, CA**	**Airbnb**

stages

stage-0	Strawman	A free-form way of submitting ideas for evolving ECMAScript. *what's required:* Create the document and add to proposal page on GitHub.
stage-1	Proposal	Create a polyfill and demos and identifying potential issues. *what's required:* At this stage, a champion who is responsible for the proposal is required. *A champion is someone approved by TC39 to take responsiblilty for the proposal.*
stage-2	Draft	A first version of what will be in the specification. *what's required:* The proposal must now additionally have a formal syntax description.
stage-3	Candidate	The proposal is mostly finished and now needs feedback from implementations and users to progress further. *what's required:* Spec text finished. Review and sign off by TC39 reviewers and the ECMAScript spec editor.
stage-4	Finished	A feature will be in the next release as the standard. *what's required:* After Test262 acceptance tests, implement to more than two browsers.

Current proposals status is here.

babel/babylon

Babylon is a JavaScript parser for Babel.
Integrate TC39's Test262 suite to Babylon.

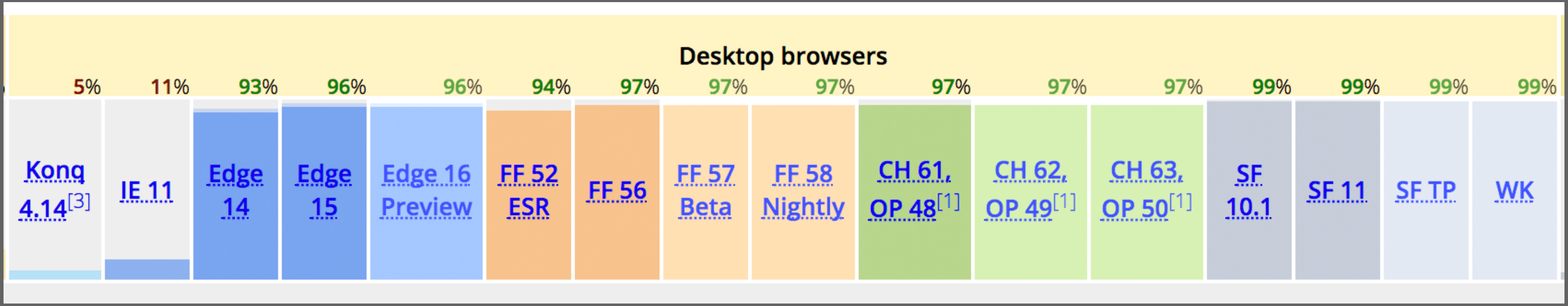
ES2015

- Arrow Functions
- Classes
- Template Strings
- Destructuring
- Default parameters
- Rest parameters
- Spread Operator
- let, const statement
- Modules
- Promise
- Generators
- for...of
- Unicode
- Map, Set, WeakMap, WeakSet
- Proxy
- Symbol
- Subclassable Built-ins
- Binary and Octal Literals
- Number, Math, Object.assign
- Array.from, Array.of
- Array.prototype.copyWithin
- Reflect
- Tail Calls

ES2015

```
class Robot {
  constructor(...args) {
    const [
      age,
      name
    ] = args;

    this.name = name;
  }

  say() {
    return `hello, ${this.name}`;
  }
}
```

Browser Compatibility



Desktop browsers																
5%	11%	93%	96%	96%	94%	97%	97%	97%	97%	97%	97%	99%	99%	99%	99%
Konq 4.14[3]	IE 11	Edge 14	Edge 15	Edge 16 Preview	FF 52 ESR	FF 56	FF 57 Beta	FF 58 Nightly	CH 61, OP 48[1]	CH 62, OP 49[1]	CH 63, OP 50[1]	SF 10.1	SF 11	SF TP	WK

We want to forget IE.😇

Tail Call Optimisation is not implemented except for Safari.(probably the priority is low)

ESM (ES Modules)

```
import { foo } from 'https://sample-xxx/foo.js';

export default function out() {}

// Node.js  e.g. main.mjs
import fs from 'fs';
// can not use `named export`
// import { readFile } from 'fs'; // error: because of `fs` is CJS
// You have to know what the import package is
import foo from './foo.mjs'; // if you use same code on both server and browser
// the browser does not automatically give extensions
```

Loader Standard in whatwg/loader.
Chrome, Firefox, Safari and Edge support ESM.
Node.js supports ESM as stability-1 from 8.5.0.🎉
But, Node.js still has many TODO because of backward compatible now.(e.g. `.mjs`)

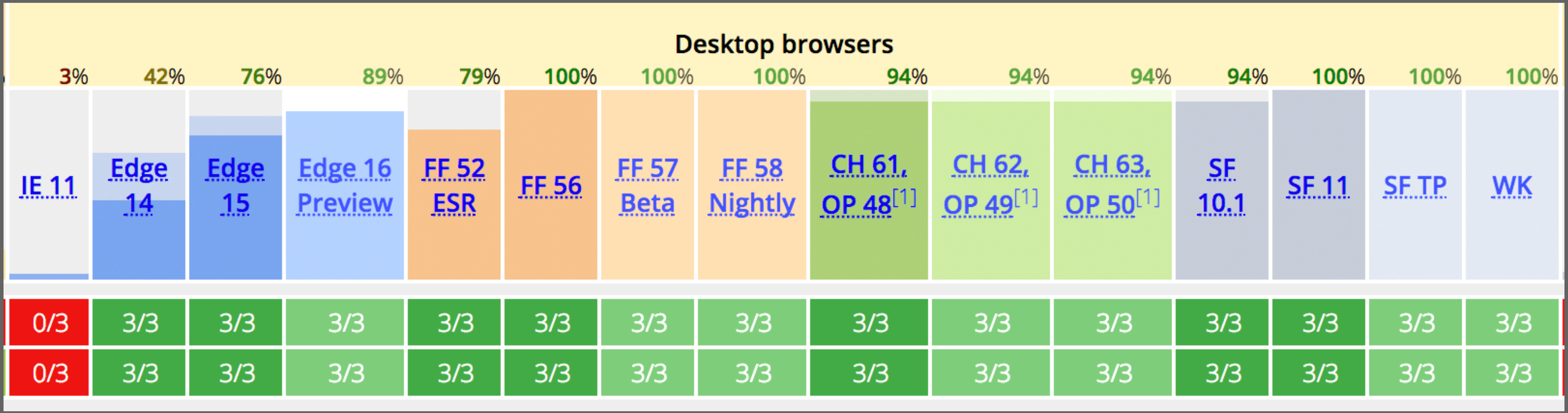
ES2016

- Array.prototype.includes
- Exponentiation Operator (**)

ES2016

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

6 ** 2 === Math.pow(6, 2);      // true
```

Browser Compatibility



Desktop browsers

IE 11	Edge 14	Edge 15	Edge 16 Preview	FF 52 ESR	FF 56	FF 57 Beta	FF 58 Nightly	CH 61, OP 48[1]	CH 62, OP 49[1]	CH 63, OP 50[1]	SF 10.1	SF 11	SF TP	WK
3%	42%	76%	89%	79%	100%	100%	100%	94%	94%	94%	94%	100%	100%	100%
0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3
0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3

Oh... IE🙄

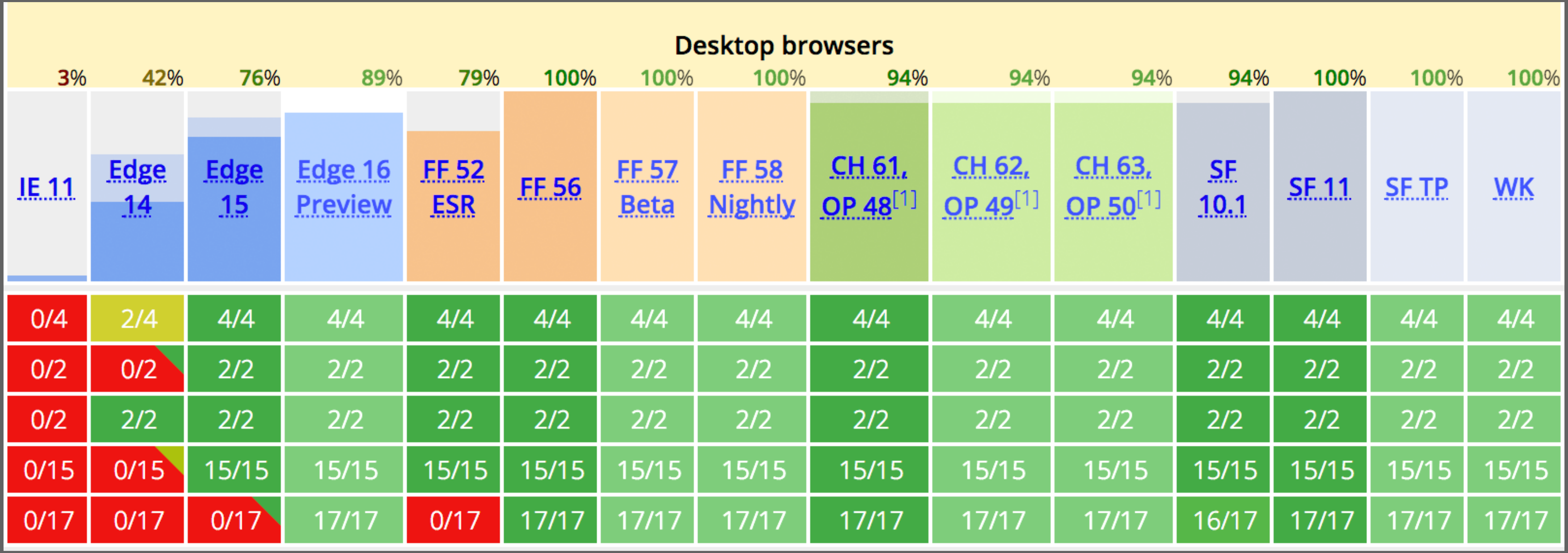
ES2017

– Async Functions
– Shared memory and atomics
– Object.values/Object.entries
– String padding
– Object.getOwnPropertyDescriptors()
– Trailing commas in function parameter lists and calls

ES2017

```
const obj = { a: 1, b: 2 };

Object.values(obj);    // [1, 2]
Object.entries(obj);   // [ [ 'a', 1 ], [ 'b', 2 ] ]
Object.getOwnPropertyDescriptors(obj);
// { a: { value: 1, writable: true, enumerable: true, configurable: true },
//   b: { value: 2, writable: true, enumerable: true, configurable: true } }

const str = 'es2017';
str.padStart(10);      // '    es2017'
str.padEnd(10);        // 'es2017    '

const fetch = (data,) => new Promise((resolve) => resolve()); // Comma is allowed

(async() => { // async/await
  await fetch({a: 1});
})();
```

Browser Compatibility



Desktop browsers

IE 11	Edge 14	Edge 15	Edge 16 Preview	FF 52 ESR	FF 56	FF 57 Beta	FF 58 Nightly	CH 61, OP 48[1]	CH 62, OP 49[1]	CH 63, OP 50[1]	SF 10.1	SF 11	SF TP	WK
3%	42%	76%	89%	79%	100%	100%	100%	94%	94%	94%	94%	100%	100%	100%
0/4	2/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
0/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
0/17	0/17	0/17	17/17	0/17	17/17	17/17	17/17	17/17	17/17	17/17	16/17	17/17	17/17	17/17

The bottom row is `Shared memory and atomics.`

Next Year is ...?

ES2018

– Template Literal Revision

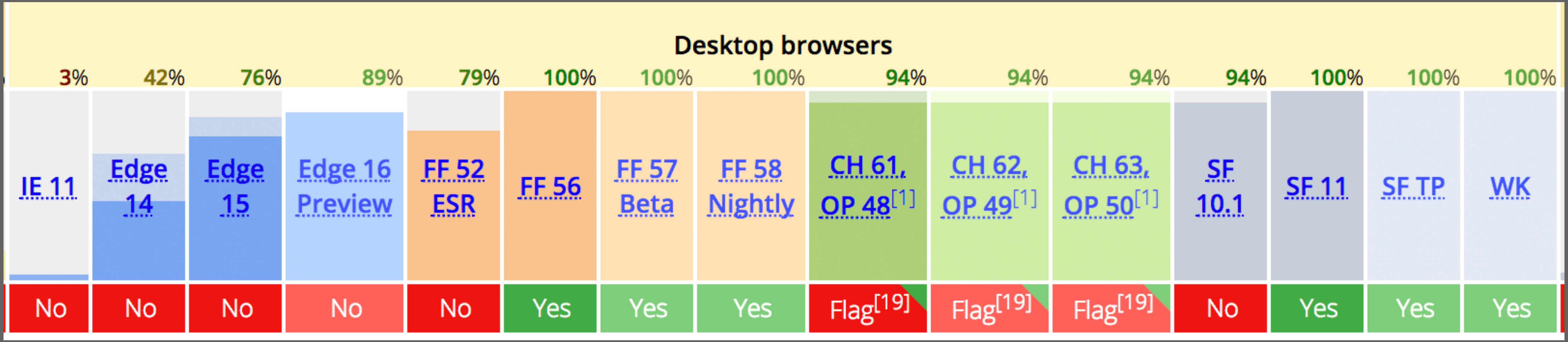
Currently, this is the only one in stage-4.

ES2018

```
const tag = (obj) => ({
  Raw: obj.raw,
  Cooked: obj
});

tag`\u{4B}`; // ES2015 ~
// { Raw: [ '\\u{4B}' ], Cooked: [ 'K' ] }

// ES2018 ~
tag`\uu ${1} \xx`; // a Unicode escape
// { Raw: [ '\\uu ', ' \\xx' ], Cooked: [ undefined, undefined ] }
tag`\100`; // an octal escape
```

Escape sequences are limited in ES2015. For example, Latex, Path, octal escape, etc...
Therefore, It will release that restriction.
An illegal escape sequence is put in the Cooked array as `undefined`.

Browser Compatibility



Desktop browsers

3%	42%	76%	89%	79%	100%	100%	100%	94%	94%	94%	94%	100%	100%	100%
IE 11	Edge 14	Edge 15	Edge 16 Preview	FF 52 ESR	FF 56	FF 57 Beta	FF 58 Nightly	CH 61, OP 48[1]	CH 62, OP 49[1]	CH 63, OP 50[1]	SF 10.1	SF 11	SF TP	WK
No	No	No	No	No	Yes	Yes	Yes	Flag[19]	Flag[19]	Flag[19]	No	Yes	Yes	Yes

Candidate Features(Stage-3)

Proposal list that may enter the next release(ES2018).

- Asynchronous Iteration
- BigInt
- Class Fields
- global
- Function.prototype.toString revision
- import()
- import.meta
- Legacy RegExp features in JavaScript
- Optional catch binding
- Promise.prototype.finally
- RegExp Lookbehind Assertions
- RegExp named capture groups
- Rest/Spread Properties
- s (dotAll) flag for regular expressions

Proposal updates

the 60th meeting on 26-28 September 2017 (about 2 weeks ago!!)

Proposal	Stage	Description
import.meta	2 → 3	
export ns from 'mod';	2 → N/A	it moved to https://github.com/tc39/ecma262/pull/1005
Array.prototype.flat{Map,ten}	1 → 2	
throw Expressions	1 → 2	
String.prototype.matchAll	1 → 2	
Extensible numeric literals	0 → 1	
First-Class Protocols	0 → 1	it was called `Interfaces` in stage-0
JSON superset	0 → 1	
nullary coalescing	0 → 1	
Partial application	0 → 1	
Pipeline Operator	0 → 1	
ArrayBuffer.transfer	N/A → 0	
Builtins.typeOf() and Builtins.is()	N/A → 0	
Object Shorthand Improvements	N/A → 0	

Proposals of Interest🙃

Introducing some proposals that I am interested in.

Note: stage-1 is subject to change

class fields

stage-3

```
class Counter extends HTMLElement {
  x = 0;  // public fields
  #y = 0; // private fields

  constructor() {
    super();
    this.onclick = this.clicked.bind(this);
  }

  clicked() {
    this.x++;
    this.#y++;
    window.requestAnimationFrame(this.render.bind(this));
  }

  render() {
    this.textContent = this.#y.toString();
  }
}
```

Optional Catch Binding

stage-3

```
let data;

try {
  data = JSON.parse(str);
} catch { //  you don't need the binding `()`
  data = 'default';
}
```

Promise.prototype.finally

stage-3

```
let finished = false;

fetch()
  .then((res) => {
    // finished = true;
  })
  .catch((err) => {
    // finished = true;
  })
  .finally(() => {
    finished = true;
  });
```

Optional Chaining

stage-1

```
// before
const fooInput = myForm.querySelector('input[name=foo]');
const fooValue = fooInput ? fooInput.value : undefined;

// after
const fooValue = myForm.querySelector('input[name=foo]')?.value;

const obj = {
  foo: {
    bar: {
      baz: 42
    }
  }
};

obj?.foo?.bar?.baz;    // 42
obj?.qux?.baz;         // undefined
obj?.foo.bar.qux?.(); // undefined
```

Pipeline Operator

stage-1

```
function doubleSay (str) { return str + ', ' + str; }
function capitalize (str) { return str[0].toUpperCase() + str.substring(1); }
function exclaim (str) { return str + '!'; }

// exclaim(capitalize(doubleSay('hello'))); // 'Hello, hello!'
let result = "hello"
  |> doubleSay // the argument is 'hello'
  |> capitalize
  |> exclaim;  // 'Hello, hello!'

function double (x) { return x + x; }
function add (x, y) { return x + y; }
function boundScore (min, max, score) { return Math.max(min, Math.min(max, score)); }

// boundScore(0, 100, add(7, double(person.score)));
let newScore = 25
  |> double                           // the argument is 25
  |> _ => add(7, _)                   // `_` is the return value of `double`
  |> _ => boundScore(0, 100, _); // 57 // `_` is the return value of `add`
```

Partial Application Syntax

stage-1

```
function add(x, y) { return x + y; }

// before
const addOne = add.bind(null, 1); // this, the left(x = 1), the right(y = undefined)
addOne(2); // 3

// after
const addOne = add(1, ?); // apply from the left(x)
addOne(2); // 3

const addTen = add(?, 10); // apply from the right(y)
addTen(2); // 12

const f = (...x) => x;
const g = f(..., 9, ...);
g(1, 2, 3); // [1, 2, 3, 9, 1, 2, 3]

const res = a |> f(?, 1) |> g(?, 2); // const res = g(f(a, 1), 2);
```

Promise.try

stage-1

```
function getUserById(id) {
  return Promise.try(() => {
    if (typeof id !== 'number') {
      throw new Error('id must be a number');
    }

    return db.getUserById(id);
  });
}
```

Temporal

stage-1

```
let civilDate = new CivilDate(year, month, day); // month 1 - 12

let year = civilDate.year;
let month = civilDate.month;
let day = civilDate.day;

new CivilTime(hour, minute[[[, second], millisecond], nanosecond]);
let hour = civilTime.hour;
let minute = civilTime.minute;
let second = civilTime.second;
let millisecond = civilTime.millisecond;
let nanosecond = civilTime.nanosecond;

let myCivilDate = new CivilDate(2016, 2, 29);
let newCivilDate = myCivilDate.plus({years: 1, months: 2});
//results in civil date with value 2017-4-28
```

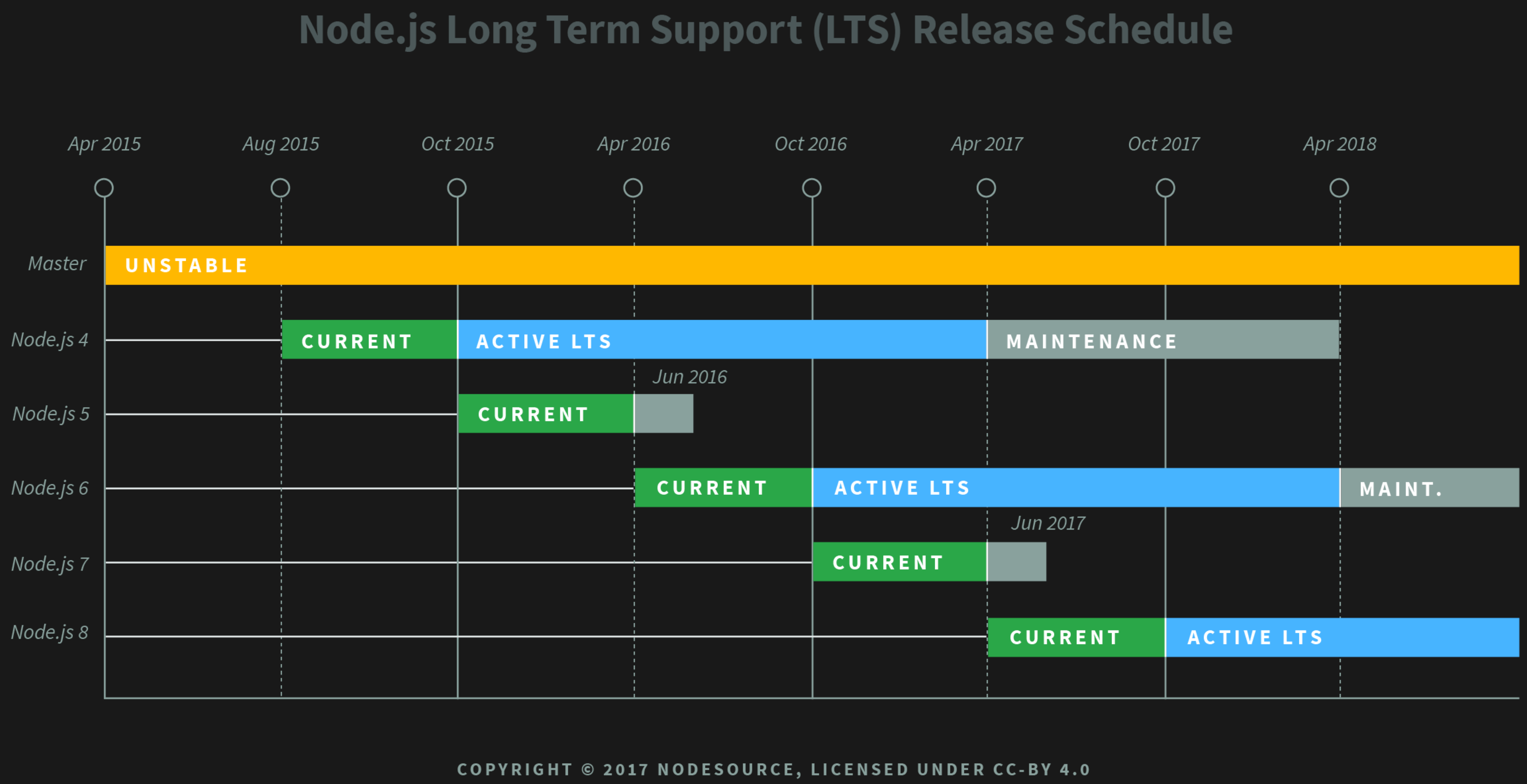
summary

- ECMAScript will be renewed every year (ES{that year})
- ECMAScript is managed by TC39 and has 5 stages
- anyone can make suggestions
- the next release will be June 2018!!

Node.js

If time is left over...😴

LTS


Node.js Long Term Support (LTS) Release Schedule
Apr 2015
Aug 2015
Oct 2015
Apr 2016
Oct 2016
Apr 2017
Oct 2017
Apr 2018
Master
UNSTABLE
Node.js 4
CURRENT
ACTIVE LTS
MAINTENANCE
Jun 2016
Node.js 5
CURRENT
Node.js 6
CURRENT
ACTIVE LTS
MAINT.
Jun 2017
Node.js 7
CURRENT
Node.js 8
CURRENT
ACTIVE LTS
COPYRIGHT © 2017 NODESOURCE, LICENSED UNDER CC-BY 4.0

LTS

A even version released in April is subject to LTS.
LTS will be supported for 18 months and then will be in maintenance mode for 12 months.

codename

- Node4: Argon
- Node6: Boron
- Node8: Carbon

Node.js Collaborator Summit

The Node.js Foundation hosts regular summits for active contributors to the node.js project.

Node.js Foundation has a lot of working groups.
570 members and 113 teams

– Oct 06-07 2017, Vancouver, British Columbia, Canada
– May 04-05 2017, Berlin, Germany
– Dec 01-02 2016, Austin, Texas, USA
– Sep 17-18 2016, Amsterdam, Netherlands
– Aug 06-07 2015, San Francisco, California, USA

information 💁‍♀️

Node Festival in Tokyo 🗼

largest Node.js conference in Japan!!!

http://nodefest.jp/2017

stuff📖

- Standard ECMA-262
- Ecma TC39
- ECMAScript Discussion Archives

THE END

ENJOY YOUR JAVASCRIPT LIFE 😎

repo: https://github.com/hiroppy/slides