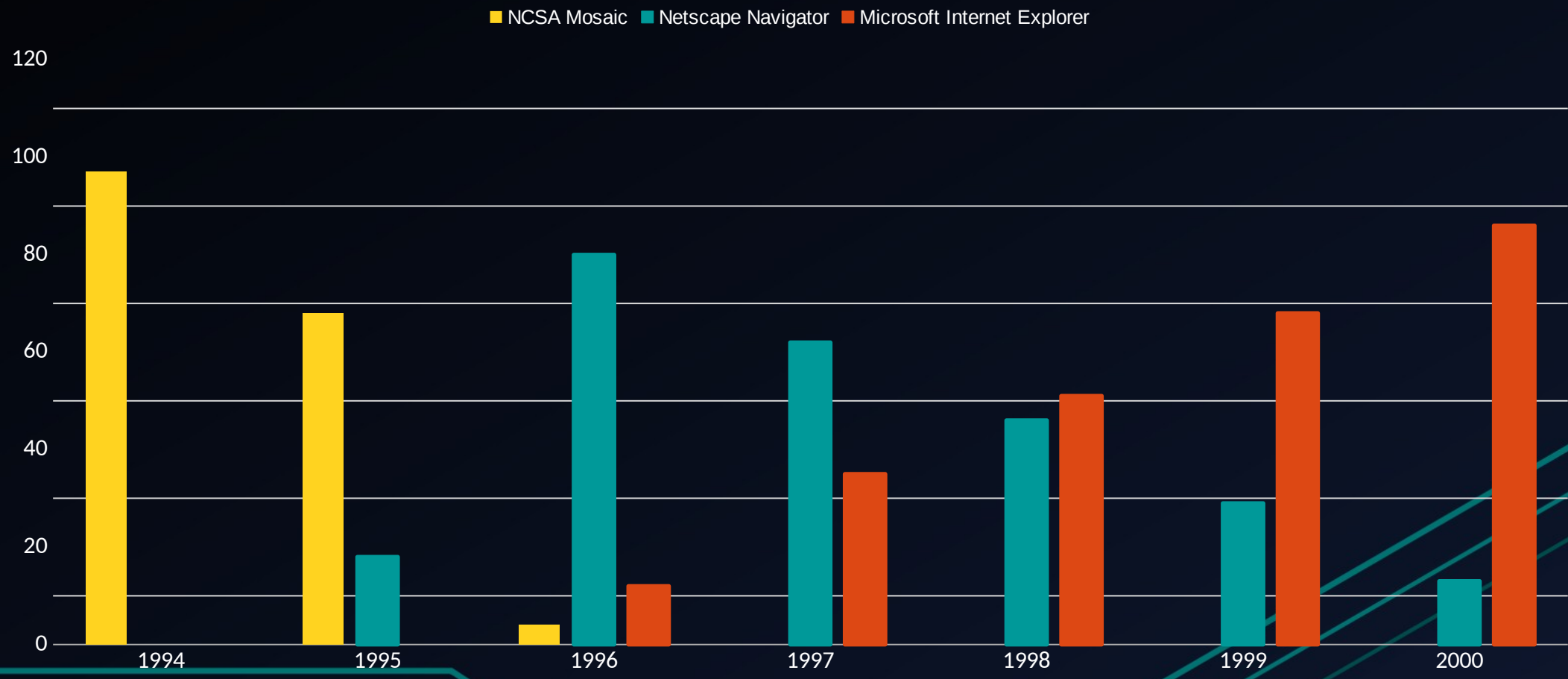
ECMAScript6 for everyone

2015 IS ALMOST THERE

by Chris Stenke <chris@isfett.com>

Browser wars (1994-2000)



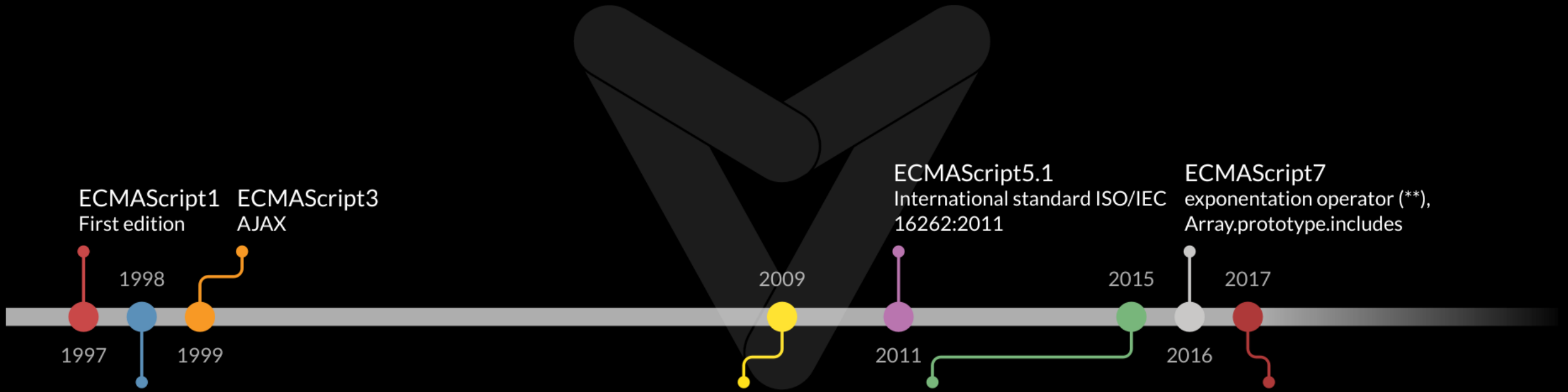
Javascript – the beginning

- Brendan Eich (33)(Netscape) wrote the prototype of JavaScript in 10 days (May 1995)
- As a matter of fact, Scheme (1970, Lips dialect) should be integrated in Netscape
- Influenced by C, Perl, Python, Java, Scheme and Hypertalk
- LiveScript got shipped in Netscape Navigator 2.0 beta 1 (September 1995), it has been renamed to JavaScript in Netscape Navigator 2.0 beta3 (December 1995) for marketing purposes
- Netscape introduced an implementation of Server-Side Javascript in Netscape Enterprise Server(December 1995)

Why ECMAScript?

- Microsoft reverse-engenieered JavaScript to create JScript for Internet Explorer 3.0 (August 1996) and Microsoft Internet Information Server (IIS)
- Microsoft added new date methods to alleviate the Year 2000 problem (Millenium-Bug) (caused by the JavaScript methods that were based on the Java Data Class), JScript got extended more and more in the next months
- JScript was not compatible to JavaScript, web developers had problemes developing websites for both browsers
- Standardization of JavaScript and JScript started by Ecma International (November 1996)
- First edition of ECMAScript was published (June 1997)

History of ECMAScript



Chris Stenke <chris@isfett.com>

ES6 Modules are starting to land in browsers

- Chrome 61
- Safari 10.1
- Firefox 54 – behind the dom.moduleScripts.enabled setting in about:config
- Edge 15 – behind the Experimental JavaScript Features setting in about:flags
- No transpilers needed anymore!

ECMAScript 5 6 2016+ next intl non-standard compatibility table

Flattr | by kangax Gratipay & webbedspace & zloirock | Fork 527

Sort by: Engine types | Show obsolete platforms | Show unstable platforms

V8 | SpiderMonkey | JavaScriptCore | Chakra | Carakan | KJS | Other

Minor difference (1 point) | Small feature (2 points) | Medium feature (4 points) | Large feature (8 points)

		Compilers/polyfills					Desktop browsers																Servers/runtimes						
	97%	56%	71%	48%	59%	17%	5%	11%	93%	96%	96%	94%	97%	97%	97%	97%	97%	97%	99%	99%	99%	99%	4%	66%	95%	59%	52%	97%	97%
Feature name	Current browser	Traceur	Babel + core-js[2]	Closure	Type-Script + core-js	es6-shim	Kong 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	PJS	Echo JS	XS6	JXA	Node 4[5]	Node >=6.5 <7[5]	Node >=8.3 <9[5]
Optimisation																													
proper tail calls (tail call optimisation)	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	0/2	0/2	2/2	0/2	0/2	0/2	0/2
Syntax																													
default function parameters	7/7	4/7	4/7	5/7	5/7	0/7	0/7	0/7	7/7	7/7	7/7	6/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	0/7	4/7	7/7	0/7	0/7	7/7	7/7
rest parameters	5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	3/5	5/5	0/5	0/5	5/5	5/5
spread (...) operator	15/15	15/15	13/15	12/15	4/15	0/15	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	15/15	0/15	10/15	15/15	11/15	0/15	15/15	15/15
object literal extensions	6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	0/6	5/6	6/6	5/6	6/6	6/6	6/6
for..of loops	9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	7/9	9/9	9/9	7/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	9/9	0/9	7/9	9/9	8/9	7/9	9/9	9/9
octal and binary literals	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	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	4/4	0/4	2/4	4/4	4/4	4/4	4/4	4/4
template literals	5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	4/5	5/5	5/5	5/5	5/5	5/5
RegExp "y" and "u" flags	5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	2/5	2/5	0/5	0/5	5/5	5/5
destructuring, declarations	22/22	20/22	21/22	20/22	15/22	0/22	0/22	0/22	21/22	22/22	22/22	21/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	0/22	12/22	21/22	19/22	0/22	22/22	22/22
destructuring, assignment	24/24	23/24	24/24	21/24	19/24	0/24	0/24	0/24	23/24	24/24	24/24	23/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	0/24	14/24	24/24	21/24	0/24	24/24	24/24
destructuring, parameters	24/24	19/24	21/24	18/24	16/24	0/24	0/24	0/24	22/24	23/24	23/24	21/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	0/24	12/24	23/24	18/24	0/24	24/24	24/24
Unicode code point escapes	2/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	1/2	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
new.target	2/2	0/2	0/2	0/2	0/2	0/2	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	2/2	0/2	2/2	2/2	0/2	0/2	2/2	2/2
Bindings																													
const	16/16	14/16	14/16	14/16	14/16	0/16	2/16	12/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	1/16	8/16	16/16	10/16	9/16	16/16	16/16
let	12/12	10/12	10/12	10/12	10/12	0/12	0/12	10/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	0/12	8/12	12/12	0/12	6/12	12/12	12/12
block-level function declaration[14]	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes
Functions																													
arrow functions	13/13	11/13	9/13	10/13	9/13	0/13	0/13	0/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	0/13	7/13	12/13	0/13	9/13	13/13	13/13
class	24/24	17/24	19/24	13/24	19/24	0/24	0/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	0/24	22/24	24/24	18/24	0/24	24/24	24/24
super	8/8	7/8	4/8	6/8	7/8	0/8	0/8	0/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	0/8	6/8	8/8	7/8	0/8	8/8	8/8
generators	27/27	24/27	24/27	16/27	0/27	0/27	0/27	0/27	27/27	27/27	27/27	25/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	27/27	0/27	16/27	27/27	0/27	20/27	27/27	27/27
Built-ins																													
typed arrays	46/46	0/46	45/46	0/46	45/46	0/46	8/46	16/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	46/46	18/46	37/46	46/46	46/46	43/46	46/46	46/46
Map	19/19	14/19	19/19	14/19	19/19	15/19	0/19	8/19	18/19	19/19	19/19	18/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	0/19	17/19	19/19	18/19	17/19	19/19	19/19
Set	19/19	14/19	19/19	14/19	19/19	15/19	0/19	8/19	18/19	19/19	19/19	18/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	19/19	0/19	18/19	19/19	18/19	17/19	19/19	19/19
WeakMap	12/12	6/12	12/12	9/12	12/12	0/12	0/12	6/12	11/12	12/12	12/12	11/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	0/12	9/12	11/12	11/12	11/12	12/12	12/12

Only pack the essentials

No need to use Webpack and Babel anymore

Module syntax

```
<script type="module">
  import {addTextToBody} from './utils.js';

  addTextToBody('Modules are pretty cool.');
</script>
```

```
// utils.js
export function addTextToBody(text) {
  const div = document.createElement('div');
  div.textContent = text;
  document.body.appendChild(div);
}
```

import

```
// Supported:
import {foo} from 'https://jakearchibald.com/utils/bar.js';
import {foo} from '/utils/bar.js';
import {foo} from './bar.js';
import {foo} from '../bar.js';

// Not supported:
import {foo} from 'bar.js';
import {foo} from 'utils/bar.js';
```

Backwards compatibility

```
<script type="module" src="module.js"></script>
<script nomodule src="fallback.js"></script>
```

Browsers that understand "type=module" should ignore scripts with a "nomodule" attribute.

Array Functions

ECMAScript 6 — syntactic sugar: reduced | traditional

```
odds  = evens.map(v => v + 1);
pairs = evens.map(v => ({ even: v, odd: v + 1 }));
nums  = evens.map((v, i) => v + i);
```

ECMAScript 5 — syntactic sugar: reduced | traditional

```
odds  = evens.map(function (v) { return v + 1; });
pairs = evens.map(function (v) { return { even: v, odd: v + 1 }; });
nums  = evens.map(function (v, i) { return v + i; });
```

Default Parameter Values

ECMAScript 6 — syntactic sugar: reduced | traditional

```
function f (x, y = 7, z = 42) {
    return x + y + z;
}
f(1) === 50;
```

ECMAScript 5 — syntactic sugar: reduced | traditional

```
function f (x, y, z) {
    if (y === undefined)
        y = 7;
    if (z === undefined)
        z = 42;
    return x + y + z;
};
f(1) === 50;
```

Destructuring Assignments

ECMAScript 6 — syntactic sugar: reduced | traditional

```
var { op, lhs, rhs } = getASTNode();
```

ECMAScript 5 — syntactic sugar: reduced | traditional

```
var tmp = getASTNode();
var op  = tmp.op;
var lhs = tmp.lhs;
var rhs = tmp.rhs;
```

Differences

- Your JavaScript ES6 modules must be served with one of the valid JavaScript MIME types, like text/javascript or application/javascript
- You need a webserver, its not working with the file:// prototocol anymore
- When you don‘t have a PHP- or Java-environment, you can simple use the npm-webserver serve
- Install serve with "npm install serve –save"
- Start the webserver with "serve ./public" for example

Let‘s compare some code ES3/ES5/ES6

- Codesamples provided at https://github.com/isfett/es5-es6-presentation
- Includes npm-webserver. Clone repo, change branch and start with "npm start"
- In branch "es5" are functions to draw some elements (boxes, circles, triangles) on a canvas
- In branch "es6" there is the same code, but in encapsulated classes, also extending superclasses
- The other branches are for promises, also for parallel running promises with faked latency

Room for improvements