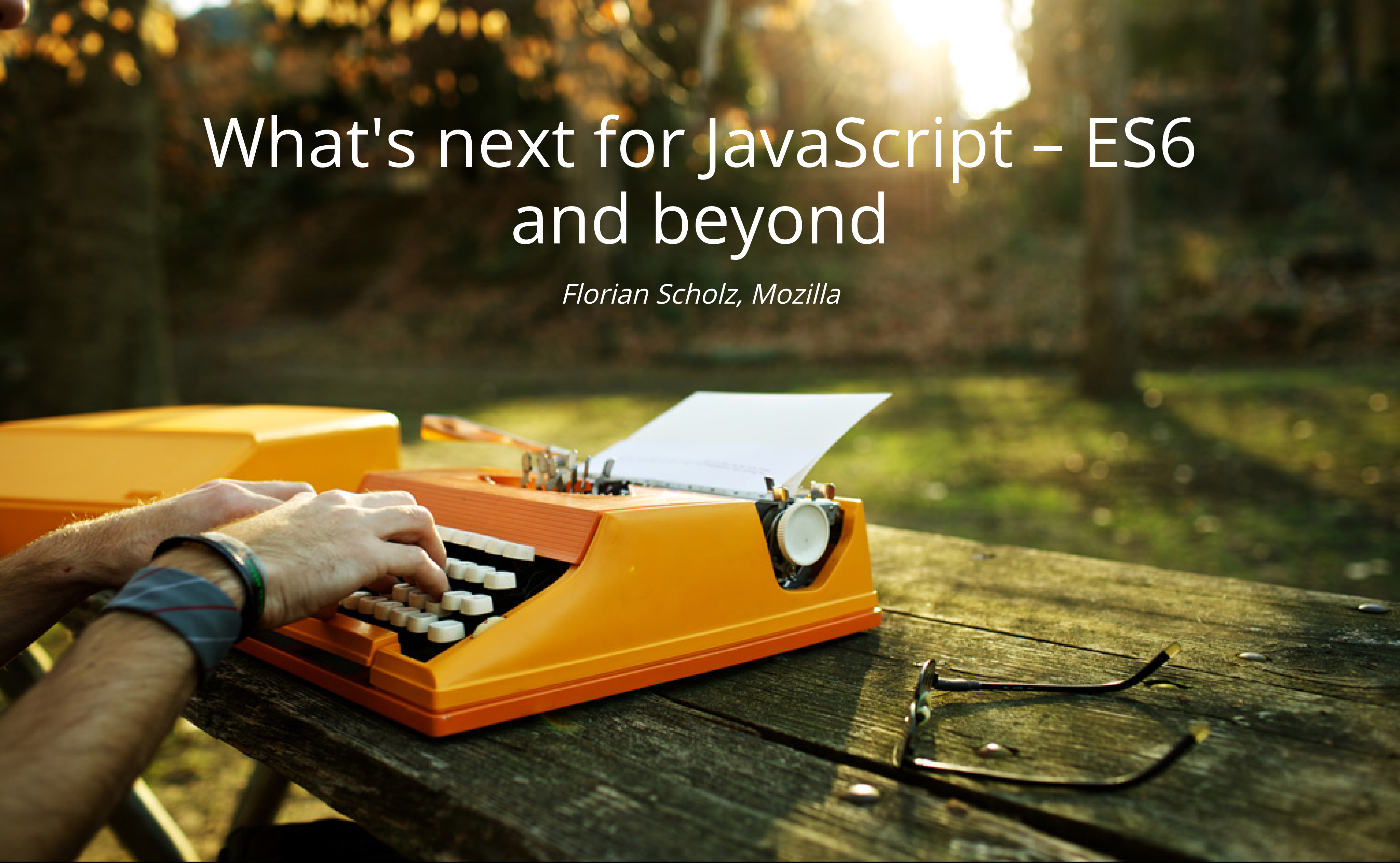
What's next for JavaScript – ES6 and beyond

Florian Scholz, Mozilla



Florian Scholz

Technical Writer

Mozilla Developer Network (MDN)

@floscholz, florianscholz.com

Brace yourselves, ES6 is coming

Standard	Pages	Release
ES 1	110	1997
ES 2	117	1998
ES 3	188	1999
ES 4	0	Never
ES 5	252	2009
ES 5.1	258	2011
ES 6	**656**	**2015 (June)**

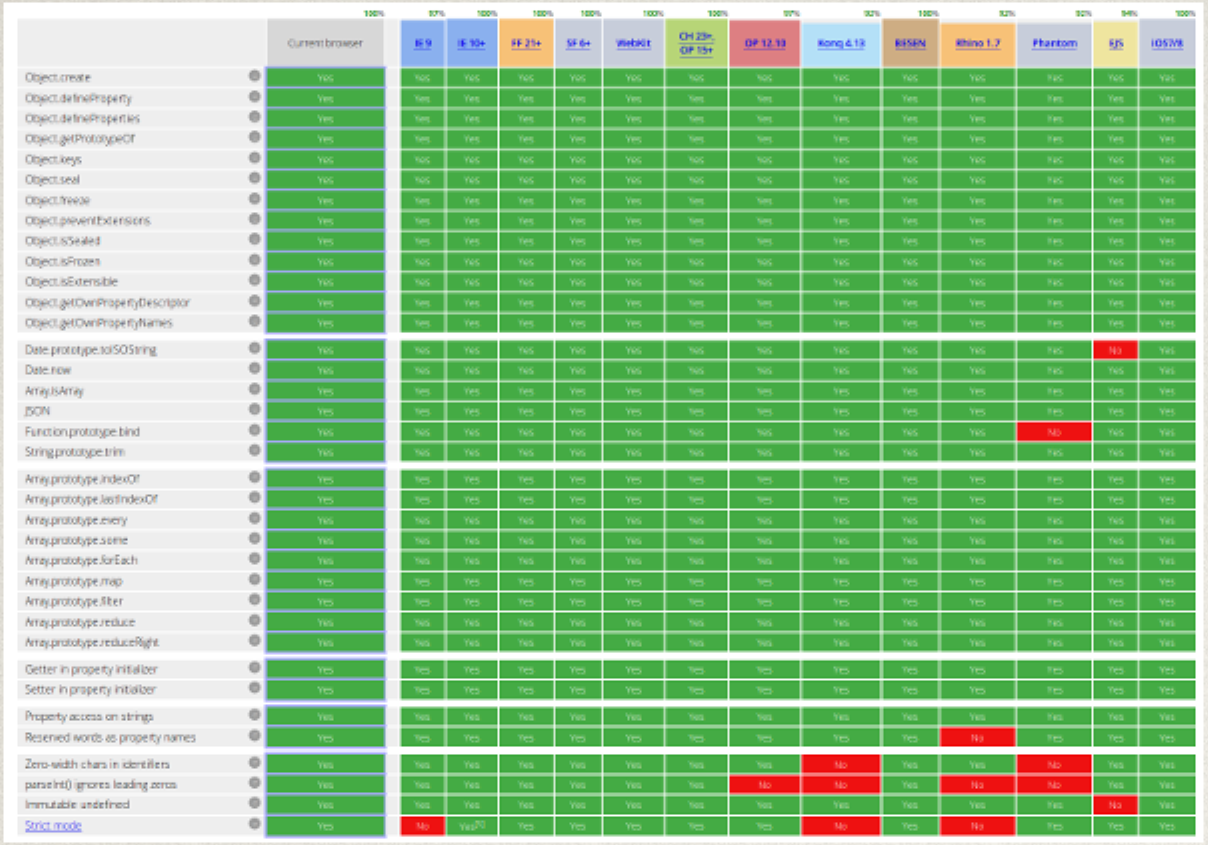
Feature history: https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript

Goals of ES6 / "harmony"

1. Make JavaScript a better language for writing
    - complex applications,
    - libraries,
    - code generators (e.g. Emscripten).
2. Improve interoperability, adopt de facto standards.
3. "Don't break the Web!". Be backwards-compatible. No versions.
4. => Also means: decisions stick, no way to remove design mistakes.

http://wiki.ecmascript.org/doku.php?id=harmony:harmony

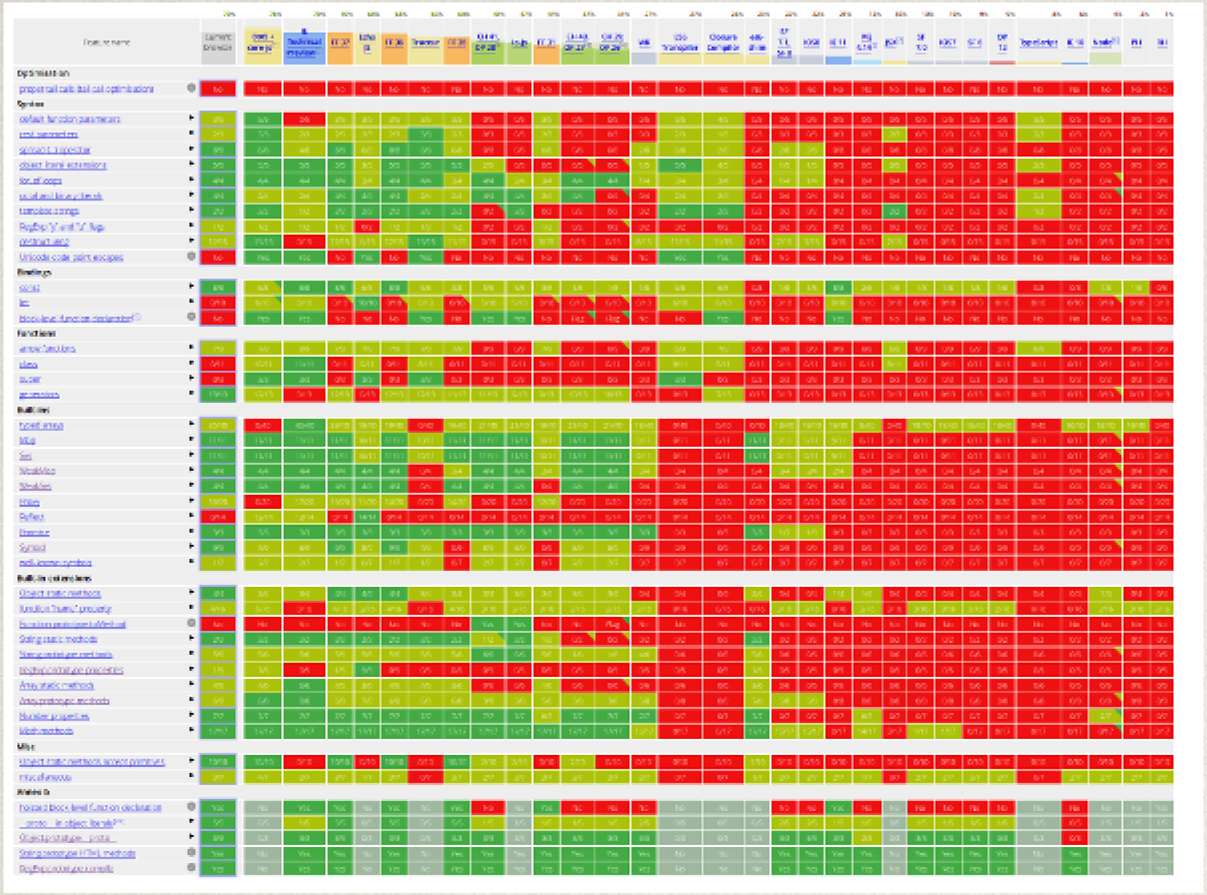
ES 5.1 implementation status



	100%	97%	100%	100%	100%	100%	100%	97%	92%	100%	92%	92%	94%	100%
	Current browser	IE 9	IE 10+	FF 21+	SF 6+	WebKit	CH 23+, OP 15+	OP 12.10	Konq 4.13	BESEN	Rhino 1.7	Phantom	EJS	iOS7/8
Object.create	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date.prototype.toISOString	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Date.now	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.isArray	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
JSON	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Function.prototype.bind	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
String.prototype.trim	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.indexOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.lastIndexOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.every	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.some	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.forEach	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.map	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.filter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.reduce	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.reduceRight	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Getter in property initializer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Setter in property initializer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Property access on strings	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reserved words as property names	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Zero-width chars in identifiers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes
parseInt() ignores leading zeros	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No	Yes	Yes
Immutable undefined	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Strict mode	Yes	No	Yes[1]	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes

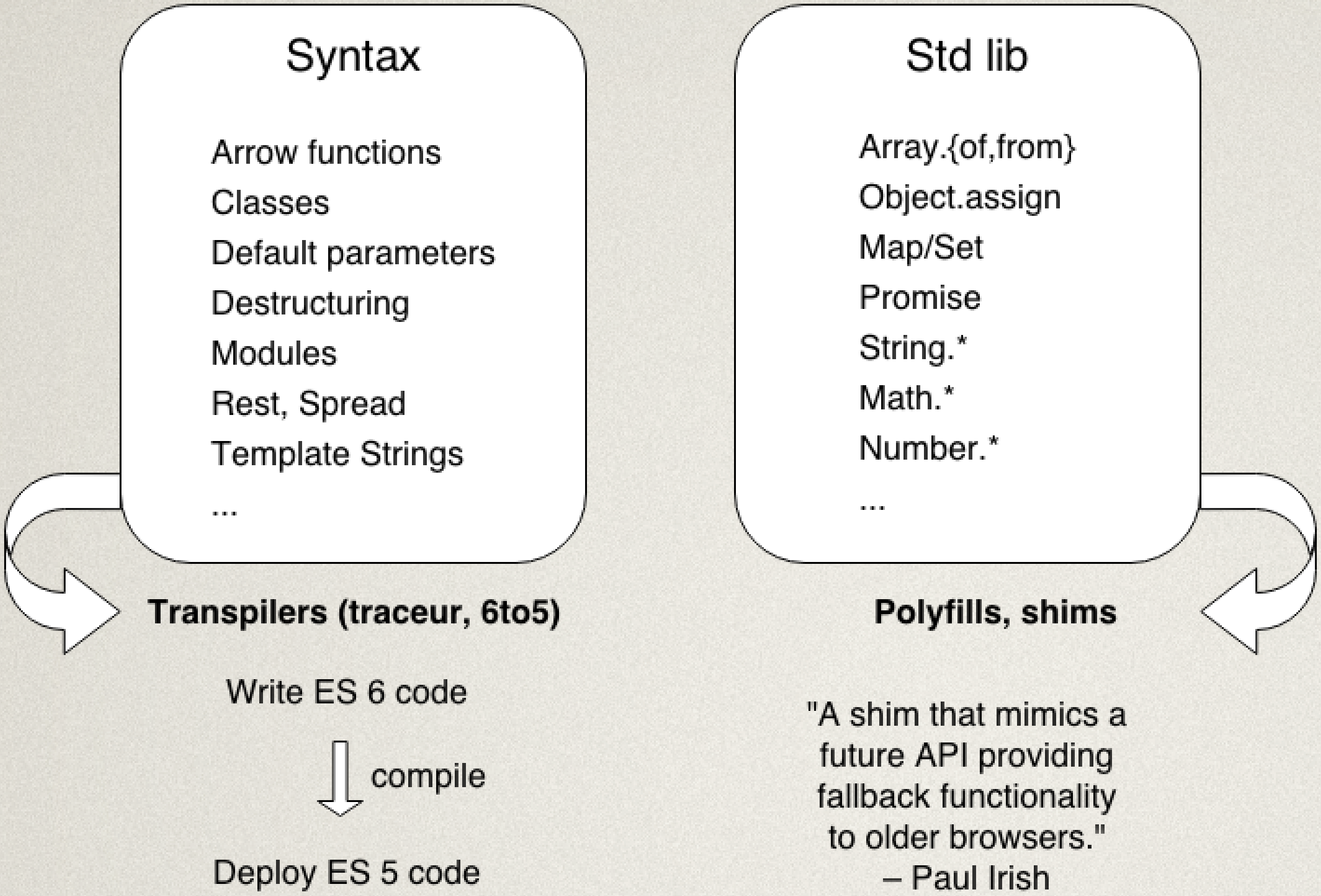
http://kangax.github.io/compat-table/es5/

ES 6 implementation status



http://kangax.github.io/compat-table/es6/

Easing into ES6 / new features



"A shim that mimics a future API providing fallback functionality to older browsers."
– Paul Irish

Enhanced syntax

Arrow functions

```
// ES 5
[1, 2, 3].map(function (x) { return x * x});
// ES 6
[1, 2, 3].map(x => x * x);
// [ 1, 4, 9 ]
```

MDN docs | Firefox 22+ | IE Spartan

Arrow functions – lexical this

```
function Person() {
  this.age = 0;
  setInterval(function growUp(){
    this.age++;
  }, 1000);
}
var p = new Person();
p.age; // 0
```

Arrow functions – lexical this

```
function Person() {
  var that = this;
  that.age = 0;
  setInterval(function growUp(){
    that.age++;
  }, 1000);
}
var p = new Person();
p.age; // 0 ... 1 ... 2 ...
```

Arrow functions – lexical this

```
function Person() {
  this.age = 0;
  setInterval(() => {
    this.age++;
  }, 1000);
}
var p = new Person();
p.age; // 0 ... 1 ... 2 ...
```

`let` and `const` – block scoping

```
function varTest() {
  var x = 31;
  if (true) {
    var x = 71;
    console.log(x); // 71
  }
console.log(x); // 71
}
```

```
function letTest() {
  let x = 31;
  if (true) {
    let x = 71;
    console.log(x); // 71
  }
console.log(x); // 31
}
```

MDN let docs | MDN const docs | Firefox | IE Spartan | (Chrome/v8) | (Safari)

Parameters: Default values

```
function multiply(a, b) {
  var b = b || 1;
  return a * b;
 }
function multiply(a, b = 1) {
  return a * b;
}
```

MDN docs | Firefox 15+

Parameters: Rest

```
function multiply(multiplier, ...theArgs) {
  return theArgs.map(x => multiplier * x);
 }

var arr = multiply(2, 1, 2, 3);
console.log(arr); // [2, 4, 6]

```

MDN docs | Firefox 15+

Spread operator

```
var parts = ['shoulder', 'knees'];
var lyrics = ['head', ...parts, 'and', 'toes'];

var arr1 = [0, 1, 2];
var arr2 = [3, 4, 5];
arr1.push(...arr2);
```

MDN docs | Firefox 16+ | IE Spartan | Safari 8

Destructuring: arrays

```
var foo = ["one", "two", "three"];
// without destructuring
var one   = foo[0];
var two   = foo[1];
var three = foo[2];
// with destructuring
var [one, two, three] = foo;
```

MDN docs | Firefox 2+ | Safari 8

Destructuring: objects

```
var o = {p: 42, q: true};
var {p, q} = o;
// shorthand for {p:p, q:q}
console.log(p); // 42
console.log(q); // true
```

MDN docs | Firefox 2+ | Safari 8

Object initializer – shorthands

```
var o = {
  drawCircle: function ([parameters]) {},
}
// ES 6
var o = {
  drawCircle([parameters]) {},
}
```

MDN docs | Firefox 34+ | IE Spartan | Chrome 41/Opera 28

Object initializer – computed properties

```
var param = 'size';
var config = {
  [param]: 12,
  ["mobile" + param[0].toUpperCase() + param.slice(1)]: 4
};
console.log(config);  // { size: 12, mobileSize: 4 }
```

MDN docs | Firefox 34+ | IE Spartan | Chrome 41/Opera 28 | Safari 8

Template Strings

```
var msg = `Hello ${document.domain}`;
// "Hello www.jfokus.se"

`multi-line
strings, auyeah`

// tagged template strings
escape`<p title="${title}">Hi ${user}!</p>`;
```

MDN docs | Firefox 34+ | IE Spartan | Chrome 41/Opera 28

Classes

```
function Person(name) {
  this.name = name;
 }
Person.prototype.introduce = function() {
  return "Hi, I am " + this.name;
};
```

Classes

```
class Person {
  constructor(name) {
    this.name = name;
  }
  introduce() {
    return "Hi, I am " + this.name
}
```

Firefox WIP | IE Spartan | Chrome WIP

Sub classes

```
class Contributor extends Person {
  constructor(name, project) {
    super(name);
    this.project = project;
  }
  introduce() {
    return super() + " and I work on " + this.project;
  }
}
```

Modules

```
// lib/utils.js
export function foo() { ... }
export const BAR = 42;
// main.js
import {foo} from 'lib/utils';
foo();
import * as utils from 'lib/utils';
utils.BAR; // 42
```

New features + Standard library

Map, Set, WeakMap, WeakSet

```
var map = new Map();
map.set(obj, "foo");
map.set("bar", "baz");

var set = new Set();
set.add(1);
set.add(obj);
set.add("foo");
```

Weak{Map,Set}:

- Keys are objects only
- Garbage Collection
- No enumeration

Map | Set | WeakMap | WeakSet | Firefox | IE Spartan | Chrome/v8 | Safari 8

for...in vs for...of

```
var arr = [ 3, 5, 7 ];
arr.foo = "hello";
for (var i in arr) {
  console.log(i); // logs "0", "1", "2", "foo"
}
for (var i of arr) {
  console.log(i); // logs 3, 5, 7
}
```

MDN docs | Firefox 13+ | IE Spartan | Chrome/v8 | (Safari 8)

Iteration – new protocols

- Iterable protocol: define iteration behavior (for `for..of`)
    - Built-ins: Arrays, Maps, Sets, Strings, Typed Arrays
    - Array-likes (e.g. DOM NodeList)
- Iterator protocol: Implementing own iterators (`next()`)

MDN docs

Generators

```
function* myGenerator() {
  for (var i = 0; i < 2; i++) {
    yield i * 2;
}}
var g = myGenerator();
g.next(); // { value: 0, done: false }
g.next(); // { value: 2, done: false }
g.next(); // { value: undefined, done: true }
```

MDN docs | Firefox | IE Spartan (WIP) | Chrome/v8

Promise – result of async operations

```
function readFile(filename, enc){
  return new Promise(function (fulfill, reject){
    fs.readFile(filename, enc, function (err, res){
      if (err) reject(err);
      else fulfill(res);
    });
  });
}
```

MDN docs | Firefox | IE Spartan | Chrome/v8 | Safari 8

Symbols – new primitive type

```
var sym = Symbol("foo");
var gsym = Symbol.for("b"); // global
typeof sym; // "symbol"
Object.getOwnPropertySymbols();

const baz = Symbol();
var obj = { [baz]: "bar" };
```

Primitives:

- Boolean
- Number
- String
- Null
- undefined
- Symbol (new)

MDN docs | Firefox 36+ | IE Spartan | Chrome/v8

Proxy – it's a trap

```
var handler = {
  get: function(target, name){
    return name in target ? target[name] : 42;
}};
var p = new Proxy({}, handler);
p.a = 1;
console.log(p.a, p.b); // 1, 42
```

MDN docs | Firefox | IE Spartan

Richer standard library

- Array: `Array.from`, `Array#fill`, `Array#find`, ...
- Math functions (`log10`, `sinh`, `fround`, `imul`, ...)
- Number: `Number.parseInt`, `Number.isSafeInteger`, ...
- Object: `Object.assign`, ..
- String: `String#fromCodePoint`, `String#includes`, ...
- Array methods for Typed Arrays (which got incorporated into ES6)
- Separate standard: `Intl` objects for Internationalization

MDN docs

ES 2016 (ES7)

Faster specification process

- Stage 1: Proposal: Driven by a champion, usage examples.
- Stage 2: Working Draft: Initial spec text, accepted by committee.
- Stage 3: Candidate Draft: Implementation experiments.
- Stage 4: Last Call Draft: Final spec, tests, 2 implementations pass.

Stage 4 proposals to be considered in a yearly edition of ECMAScript.

https://github.com/tc39/ecma262

Comprehensions (deferred from ES6)

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

[for (year of years) if (year > 2000) year];

(for (letters of ["A", "B", "C"]) letters.toLowerCase());
// generator yielding "a", "b", and "c"
```

MDN docs | Firefox

Stage 2: Object.observe()

```
var obj = { foo: 0; bar: 1; };
Object.observe(obj, function(changes) {
  console.log(changes);   }
obj.baz = 2;
 // [{name: 'baz', object: <obj>, type: 'add'}]
obj.foo = 42;
 // [{name: 'foo', object: <obj>, type: 'update',
       oldValue: 0}]
```

MDN docs | Chrome/v8

SIMD types

Single Instruction Multiple Data (Vector processing)

```
var a = SIMD.float32x4(1.0, 2.0, 3.0, 4.0);
var b = SIMD.float32x4(5.0, 6.0, 7.0, 8.0);
var c = SIMD.float32x4.add(a,b);
// [6.0, 8.0, 10.0, 12.0]
```

WebGL, Audio, codecs, Crypto, ...

https://github.com/johnmccutchan/ecmascript_simd | Firefox Nightly Channel

More

- Typed Objects
- Async generators
- Exponentiation operator: `x ** y` (same as `Math.pow(x,y)`)
- `Array#includes`, `ArrayBuffer.transfer`
- `String#at`, `String#lpad`, `String#rpad`
- More current proposals

Thanks!

developer.mozilla.org

@floscholz