
JAVASCRIPT

Head of Engineering @ eMoteev

former MTI && ACU @ EPITA

Intro

QUENTIN PRÉ

This is a just a text holder in which you assume that I wrote a lot of interesting things, which obviously I have not.
If you can’t read it easily, you are too far away, get closer to the scene.
If you still have difficulties reading this, raise your hand and ask your question.

RULES

You are encouraged to make mistakes,
You are *forbidden* to make faults.

EFFORT

Do not expect knowledge to fall into your hands.

FOCUS

If you do not intend to follow the lecture, assume consequences for your actions and stay home.

You don’t need your laptop outside of tutorials.

ASK

No question makes you stupid,
Asking no questions though…

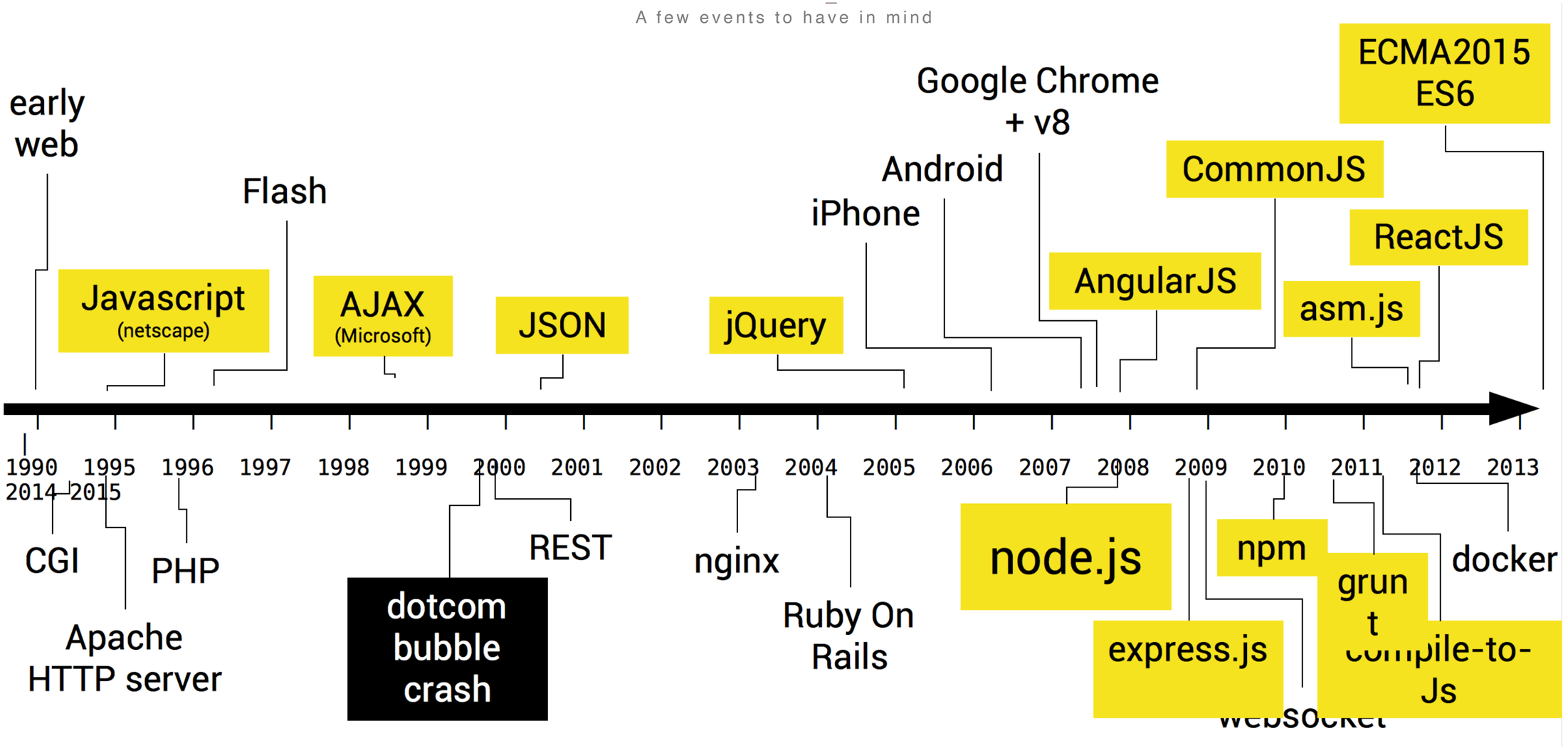
note: your question might get answered by another question.

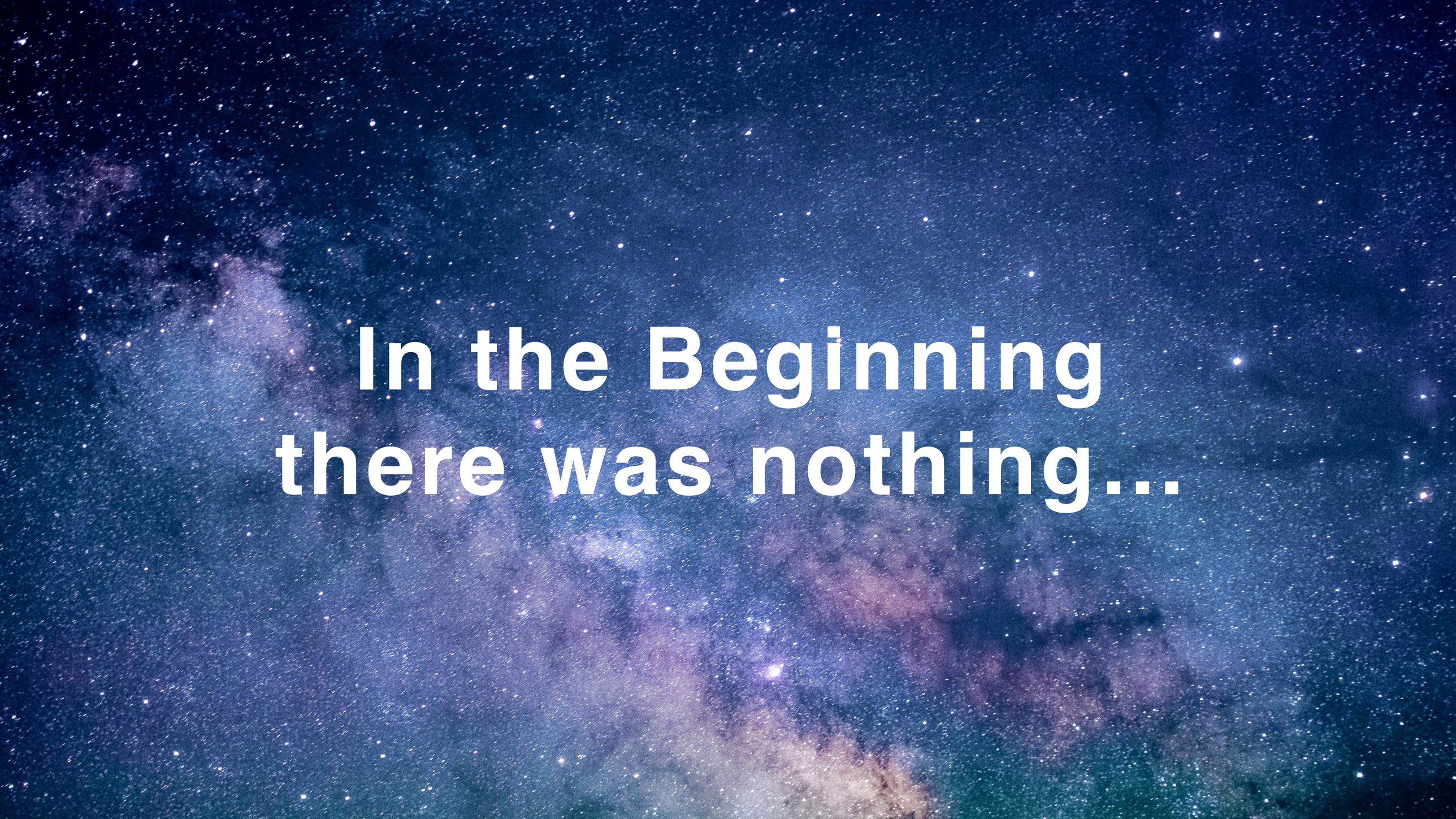
A LITTLE BIT OF HISTORY

History

The Web

A few events to have in mind




In the Beginning
there was nothing...

Then…

BRENDAN EICH

Took 10 days in 1995 at Netscape to build…

Mocha !

wait…

LiveScript ?

Oh no…

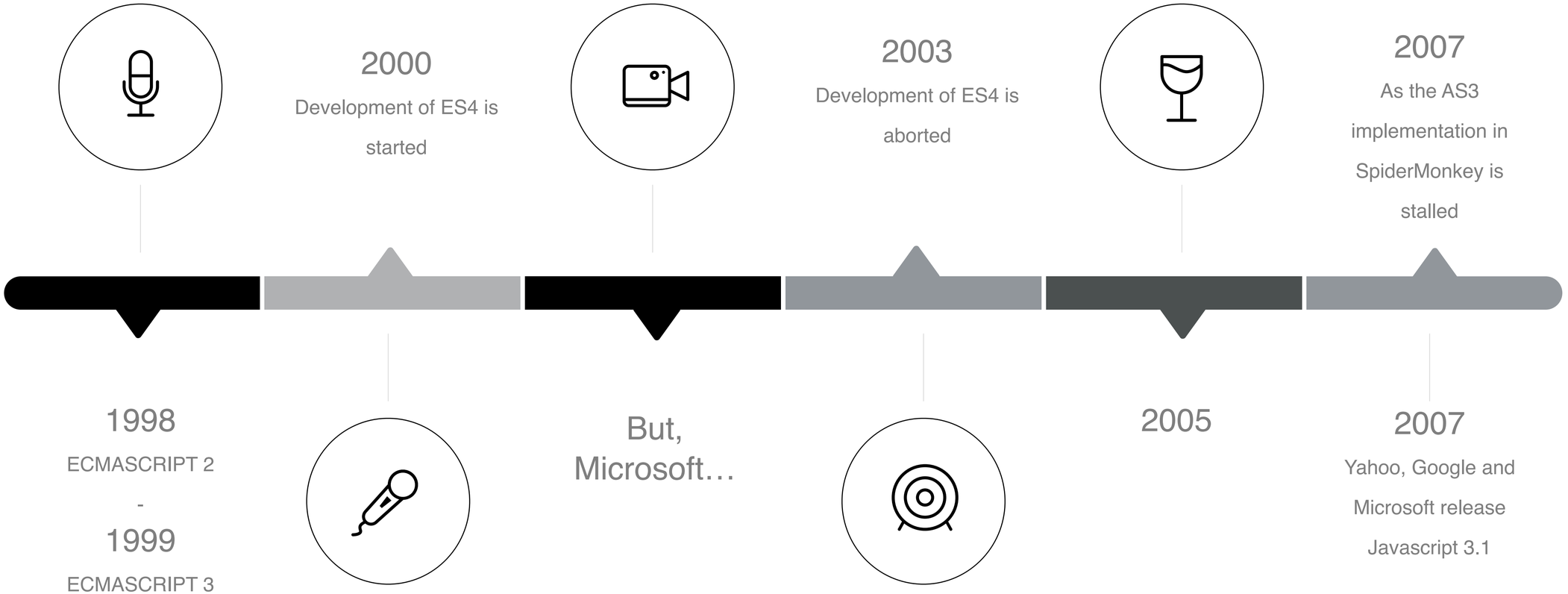
JavaScript

(because “Java” was the most successful language at the time…)

PREHISTORY

PRE-ES6

A brief history of the multiple branches of Javascript



TC-39

A COMITTEE

Made of open source contributors and engineers from browser manufacturers and major web actors (Mozilla, Apple, Microsoft, Google…)

SPECIFIYNG

They identify needs, write the specs for ECMA-262 and maintain and update this specifications.

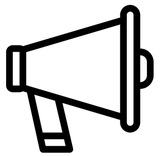
OVERVIEWING

The implementation of the standard

JOURNEY OF A FEATURE REQUEST FOR ECMA-262

Stage 0 (Strawman)

A Contributor or Member of TC39 thinks it's a good idea and writes a first document describing the feature spec

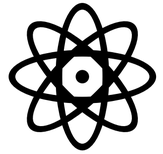


STAGE 1 (proposal)

TC-39 agrees the idea is worth pursuing

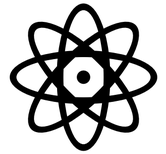
STAGE 2 (draft)

A First revision of the feature spec is written, 2 experimental implementations are launched



STAGE 3 (candidate)

Finishing feature spec, and gets enriched with feedbacks from implementations and users.



STAGE 4 (Finished)

The feature specification is finished,
The feature will be an official part of the language on the next revision of ECMA-262

ES-5

2009 - 2015

ES6 and Beyond

ES6 === ES2015

ES7 === ES2016

ES8 === ES2017

...

BrendanEich ✓
@BrendanEich

Following

Replying to @kentcdodds

It's so easy: just subtract 2009 from the year number to get the edition number.

And Behold, The TRUTH FROM ECMA!

RETWEETS 5 LIKES 14

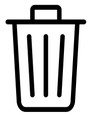
12:55 AM - 31 Mar 2017

Javascript 101

Functional

Javascript has all the paradigms for functional programming

Dynamic



Object-oriented

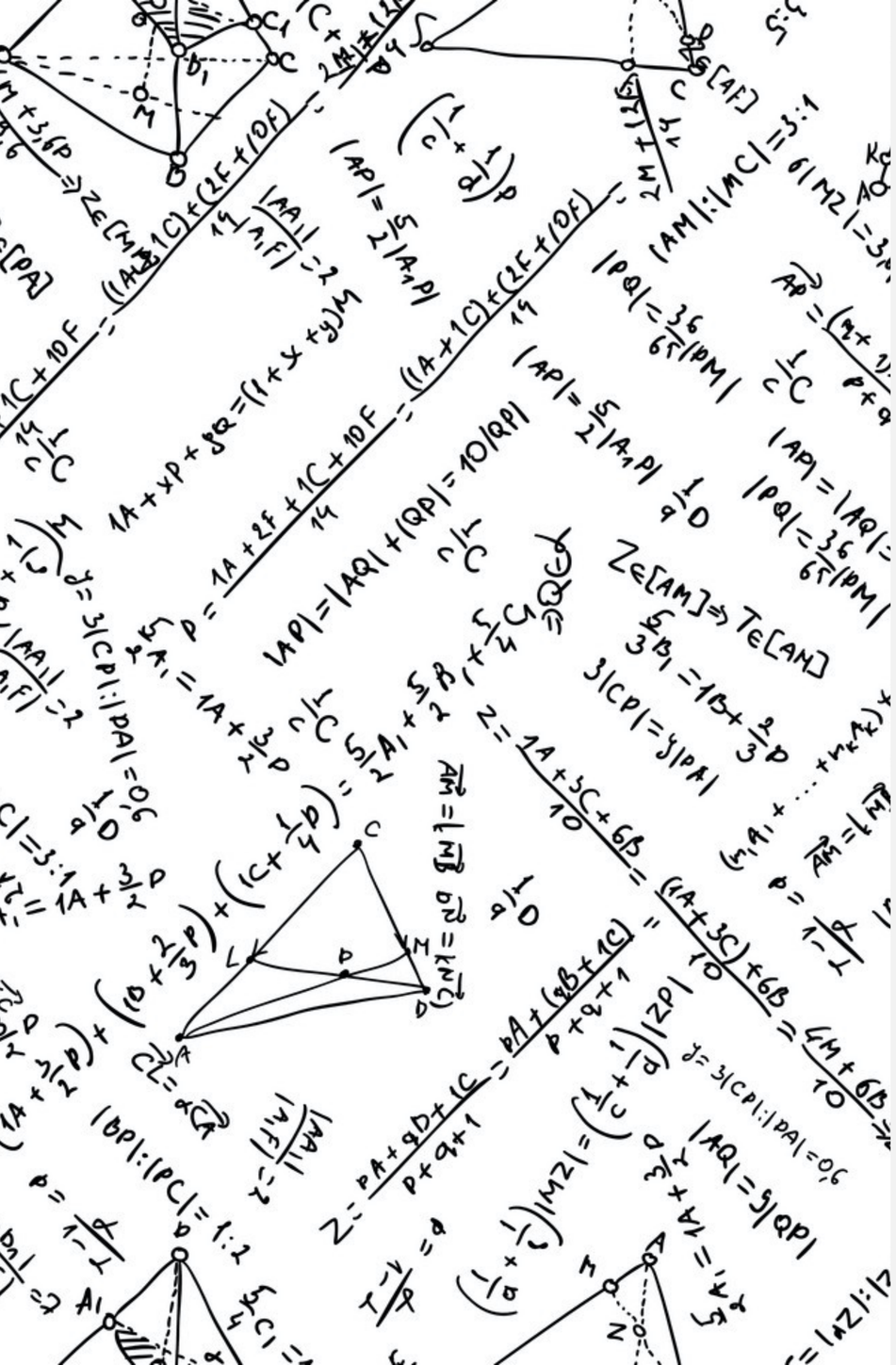
Javascript offers OOP paradigms via Prototype inheritance

Just-In-Time

Browsers will compile JS to bytecode ahead of time, allowing for performance boosts

Everywhere

Javacript is present in the browser, on the server, in IoT, in watches…



Types

Number

- Double-precision on 64-bits values
- No **Integer** type
- Math module provides common functions (cos, sin…) and constants (PI)
- *NaN* for Not A Number -> hint: avoid it
- *parseInt(value: String, base: Number) -> Number*
  careful: *parseInt('toto', 10)* returns *NaN*
- 0.1 + 0.3 -> WTF.js

Types

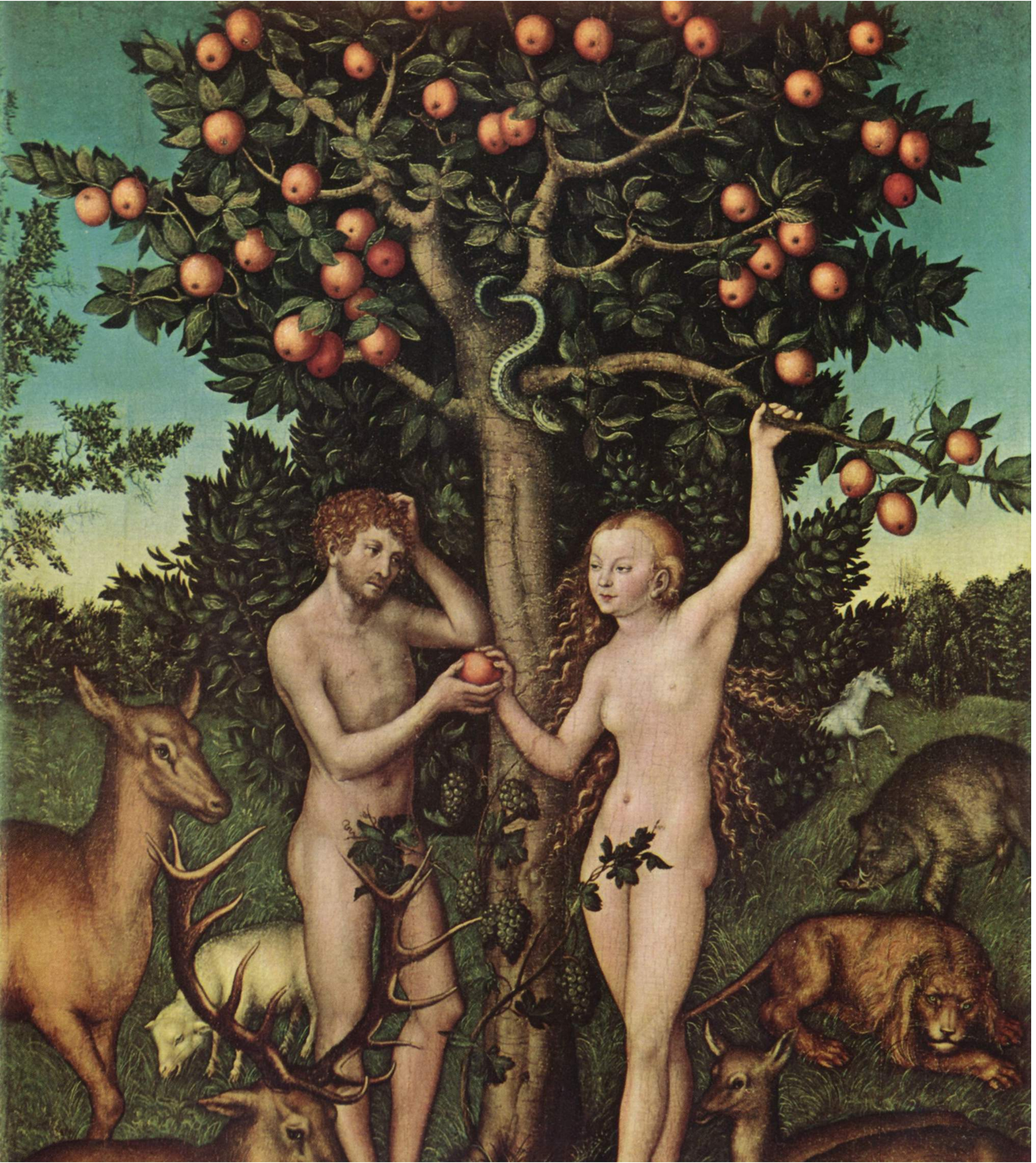
String

- Sequence of Unicode Characters
- *'Hello'.chatAt(0)* -> 'h'
- *'Hello World'.replace('hello', 'goodbye')* -> 'goodbye world'
- *'hello'.toUpperCase()* -> 'HELLO'
- Multiple methods to manipulate them -> MDN

Types

Boolean

- Values are silently converted to their Boolean equivalent when needed
- *false*, *0*, *''* (empty string), *NaN*, *null* and *undefined* all evaluate to *false* when comparing
- All other values evaluate to *true*



Types

Object

- Function
- Array
- Date
- RegExp

Types

Object

- *{ key: value }*
- *const obj = new Object()*
- *const obj = {}*
- *Obj.firstLevel.secondLevel*
- *Obj[‘firstLevel’][‘secondLevel’]*

Types

Array

- *const arr = new Array()*
- *const arr = []*
- *Map, filter, reduce …*
  are your best friends when dealing with arrays
- Multiple methods to manipulate them
  -> MDN

Types

Function

- *function getName(name) { return name; }*
- *const getName = (name) => { return name; }*
- *const getName = (name) => name;*

Types

Prototype

```
function Student(name) {
  this.name = name;
}

Student.prototype.greet = function () {
  console.log(`Hello ${this.name} !`)
}

const toto = new Student('toto');

toto.greet();
```

```
(Student.greet.bind(toto))()
```

Sugar

Class

```
class Student {
  constructor(name) {
    this.name = name;
  }

  greet() {
    console.log(`Hello ${this.name} !`)
  }
}

const toto = new Student('toto');

toto.greet();
```

Noticed the template strings ?

Scope

- **Global Scope:** Any variable declared outside of a function is in the global scope

- **Function Scope:** Any variable declared in a function can only be accessed from this function and its nested functions.
  keyword: *var*

- **Block Scope:** Introduced with ES2015, any variable declared in a block (hint: {}), can only be accessed from this block and its nested blocks.

  keyword: *let*
  keyword: *const* same as *let* except it is read-only (warning, this only applies to the stored reference…)

- **Read me:** What const really stands for in ES6

```
var foo = 'bar';

console.log(foo); // => 'bar'

function () {
  var toto = 'tata';

  console.log(toto); // => 'tata'
  console.log(foo); // => 'bar'
}

console.log(toto); // => Uncaught ReferenceError: toto is not defined
```

```
(function iife() {
  console.log('toto');
})()

iife(); // => ?
```

```
for (var i = 0; i <= 10; i++) {
  console.log(i); // => 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
}

console.log(i); // => 11
```

Let is block-scoped


```
for (let i = 0; i <= 10; i++) {
  console.log(i); // => 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
}

console.log(i); // => Uncaught ReferenceError: i is not defined
```

Const is read-only


```
const toto = 'foo';

toto = 'bar'; // => Uncaught TypeError: Assignment to constant variable
```

Well, almost read-only


```
const foo = { one: 'one' };

foo.two = 'two'; // => { one: 'one', two: 'two' }

foo.one = 'un'; // => { one: 'un', two: 'two'}
```


MAKE
LET
NOT
VAR
JS

Side Effect

Closure

A closure is the combination of a function’s scope and the captured surrounding scope scope.

```
function makeAdder(x) {
  return function(y) {
    return x + y;
  };
}

var add5 = makeAdder(5);
var add10 = makeAdder(10);

console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

Hoisting

The interpreter will treat declarations first and then follow assignments and executions

```
function fun() {
  foo(); // TypeError "foo is not a function"
  bar(); // "bar"

  var foo = function () { console.log('foo'); };
  function bar() { console.log('bar'); };
}

// is tnterpreted?

function fun() {
  var foo;
  function bar() { console.log('bar'); };

  foo();
  bar();

  foo = function () { console.log('foo'); };
}
```

Modern Javascript Useful Features

Operator

Rest/Spread

- … <= yes, this is an operator
-
```
const arr1 = [1, 2, 3, 4];
const arr2 = [4, 5, 6, 7];

const arr3 = [...arr1, ...arr2] ;
```
-
```
const add = (w, x, y, z) => {
  return Array
    .from(arguments)
    .reduce((a, b) => a + b, 0);
}

add(...arr3)
```

Rest/ Spread

-
```
const hasFoo = { foo: 'foo' };
const hasBar = { bar: 'bar' };

const hasFooAndBar = {
  ...hasFoo,
  ...hasBar
};
```
- Noticed the `argument` keyword ?

Destructuration

```
const getState = () => ({
  name: 'square',
  width: 150,
  height: 150
})

const logWidth = () => {
  const { width } = getState();
  console.log(width);
}
```

Default Params

```
const area = ({ width: 300, height: 400 })
=> {
  return width * height;
}
```

```
area({ width: 100 }); // => 40 000
```

Use it ! Using destructuring on your functions parameters will allow you (and your text editor) to always know what is needed to call your function and trace what not used anymore inside the function !

Noticed the implicit object return ?

Promise

- An object wrapping an asynchronous task

```
const prom = new Promise(
  (resolve, reject) => {
    setTimeout(() => resolve('toto'), 3000));
  }
);

prom
  .then((text) => console.log(text));
```

- In the above example, `toto` will be logged only if the promise has resolved. In this case at least after 3 seconds

- You can attach an infinity of callbacks to the promise using *.then(callback)* on the returnees object

- Rejections are handled through *.catch(cb)*

Async/Await

- Await inside an async function will lock execution until resolution

```
const getProm = () => new Promise(
  (resolve, reject) => {
    setTimeout(() => resolve('toto'), 3000));
  }
);

const log = async () => {
  const stringToLog = await getProm();
  console.log(stringToLog);
}
```

Slides
under maintenance

Modules

ES Modules

```
// lib.js

const PI = 3.14; // inaccessible from outside

export const add = (x, y) => x + y;
export const sub = (x, y) => x - y;

export default add;


// index.js

import * as Lib from 'lib';

console.log(Lib.add(2, 3)); // => 5

console.log(Lib.default(2, 3)); // => 5
```

ES Modules

```
// lib.js

const PI = 3.14; // inaccessible from outside

export const add = (x, y) => x + y;
export const sub = (x, y) => x - y;

export default add;


// index.js

import * as Lib from 'lib';

console.log(Lib.add(2, 3)); // => 5

console.log(Lib.default(2, 3)); // => 5
```

ES Modules

```
// lib.js

const PI = 3.14; // inaccessible from outside

export const add = (x, y) => x + y;
export const sub = (x, y) => x - y;

export default add;


// index.js

// remember destructuring variables ?
import { sub } from 'lib';

console.log(sub(2, 3)); // => -1
```

ES Modules

```
// lib.js

const PI = 3.14; // inaccessible from outside

export const add = (x, y) => x + y;
export const sub = (x, y) => x - y;

export default add;


// index.js

import foo from 'lib';

console.log(foo(2, 3)); // => 5
```

Common JS

```
// lib.js

const PI = 3.14; // inaccessible from outside

const add = (x, y) => x + y;
const sub = (x, y) => x - y;

module.exports = {
  add,
  sub,
};

// index.js

const { add } = require('lib');

console.log(add(2, 3)); // => 5
```

Transpilation

intl non-standard compatibility table

Flattr 59

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%	18%	5%	11%	83%	93%	95%	86%	94%	97%	97%	97%	97%	97%	97%	97%	100%	100%	100%	100%	4%	66%	96%	59%	52%	97%
Current browser	Traceur	Babel + core-js[2]	Closure	Type-Script + core-js	es6-shim	KQ 4.14[3]	IE 11	Edge 13[4]	Edge 14[4]	Edge 15[4]	FF 45 ESR	FF 52	FF 53 Beta	FF 54 Aurora	FF 55 Nightly	CH 57, OP 44[1]	CH 58, OP 45[1]	CH 59, OP 46[1]	SF 10	SF 10.1	SF TP	WK	PJS	Echo JS	XS6	JXA	Node 4[5]	Node >=6.5 <7[5]
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	0/2	2/2	2/2	2/2	2/2	0/2	0/2	2/2	0/2	0/2	0/2
7/7	4/7	4/7	4/7	5/7	0/7	0/7	0/7	0/7	7/7	7/7	4/7	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
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	5/5	0/5	3/5	5/5	0/5	0/5	5/5
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	15/15	0/15	10/15	15/15	11/15	0/15	15/15
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	6/6	0/6	5/6	6/6	5/6	6/6	6/6
9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	7/9	7/9	9/9	7/9	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
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	4/4	0/4	2/4	4/4	4/4	4/4	4/4
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	5/5	0/5	4/5	5/5	5/5	5/5	5/5
5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	2/5	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
22/22	20/22	21/22	19/22	15/22	0/22	0/22	0/22	0/22	21/22	22/22	19/22	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
24/24	23/24	24/24	17/24	19/24	0/24	0/24	0/24	0/24	23/24	24/24	21/24	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
23/23	19/23	20/23	18/23	15/23	0/23	0/23	0/23	0/23	22/23	23/23	18/23	20/23	23/23	23/23	23/23	23/23	23/23	23/23	23/23	23/23	23/23	23/23	0/23	12/23	23/23	18/23	0/23	23/23
2/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	1/2	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	0/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	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
16/16	14/16	14/16	14/16	14/16	0/16	2/16	12/16	12/16	16/16	16/16	12/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	1/16	8/16	16/16	10/16	9/16	16/16
12/12	10/12	10/12	10/12	10/12	0/12	0/12	10/12	10/12	12/12	12/12	10/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	0/12	8/12	12/12	0/12	6/12	12/12
Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes
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	13/13	0/13	7/13	12/13	0/13	9/13	13/13
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	24/24	0/24	22/24	24/24	18/24	0/24	24/24
8/8	7/8	4/8	5/8	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	8/8	0/8	6/8	8/8	7/8	0/8	8/8
27/27	24/27	24/27	16/27	0/27	0/27	0/27	0/27	27/27	27/27	27/27	25/27	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
46/46	0/46	45/46	0/46	45/46	0/46	8/46	16/46	44/46	46/46	46/46	42/46	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
19/19	14/19	19/19	14/19	19/19	15/19	0/19	8/19	18/19	18/19	19/19	18/19	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	14/19	19/19	14/19	19/19	15/19	0/19	8/19	18/19	18/19	19/19	18/19	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
12/12	6/12	12/12	9/12	12/12	0/12	0/12	6/12	11/12	11/12	12/12	10/12	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
11/11	5/11	11/11	8/11	11/11	0/11	0/11	0/11	10/11	10/11	11/11	9/11	10/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	11/11	0/11	9/11	10/11	10/11	10/11	11/11
34/34	0/34	0/34	0/34	0/34	0/34	0/34	0/34	34/34	34/34	34/34	30/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	34/34	0/34	27/34	34/34	0/34	0/34	34/34
20/20	0/20	14/20	14/20	18/20	14/20	0/20	0/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	20/20	0/20	14/20	16/20	0/20	0/20	20/20
8/8	4/8	8/8	7/8	8/8	7/8	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	8/8	0/8	4/8	8/8	0/8	7/8	8/8
12/12	4/12	9/12	2/12	8/12	2/12	0/12	0/12	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	12/12	0/12	4/12	12/12	12/12	10/12	12/12
25/26	1/26	14/26	1/26	15/26	0/26	0/26	0/26	9/26	10/26	14/26	8/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	26/26	0/26	23/26	25/26	22/26	3/26	26/26
4/4	3/4	3/4	2/4	3/4	2/4	1/4	1/4	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	4/4	0/4	4/4	4/4	4/4	4/4	4/4
17/17	0/17	8/17	0/17	3/17	0/17	3/17	0/17	14/17	16/17	16/17	10/17	12/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	3/17	5/17	17/17	14/17	6/17	17/17
2/2	2/2	2/2	1/2	2/2	2/2	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	2/2	0/2	2/2	2/2	2/2	2/2	2/2
10/10	8/10	9/10	7/10	9/10	7/10	0/10	0/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10	9/10	10/10	10/10	10/10	10/10

- Transforms ES6 into ES5 through plugins
- Each revision of ES comes as a preset of feature syntax and transformations
- You can use features before they get into the specification (warning: if a feature hasn’t reached stage-4 it might never end up in the spec)
- Polyfills missing parts of older browsers (IE8…)

BABEL


BABEL
Evaluate
Presets: es2015, react, stage-2
Line Wrap
Minify (Babili)
Babel 6.24.0
const mul2 = a => a * 2;
const arr = [1, 2, 3, 4];
const doubles = arr.map(item => mul2(item));
console.log(...doubles);
"use strict";
var _console;
function _toConsumableArray(arr) { if (Array.isArray(
var mul2 = function mul2(a) {
return a * 2;
};
var arr = [1, 2, 3, 4];
var doubles = arr.map(function (item) {
return mul2(item);
});
(_console = console).log.apply(_console, _toConsumabl

-

```
{
  "presets": [
    ["env", {
      "targets": {
        "browsers": ["last 2 versions", "safari >= 7"]
      }
    }]
  ]
}
```

- Now that most modern browsers support most ES6 features, only transforms what you need for your audience.

JS as a platform

- TypeScript
- CoffeeScript
- ReasonML
- ClojureScript
- …
- Nowadays almost every existing language has a way of being compiled to JS

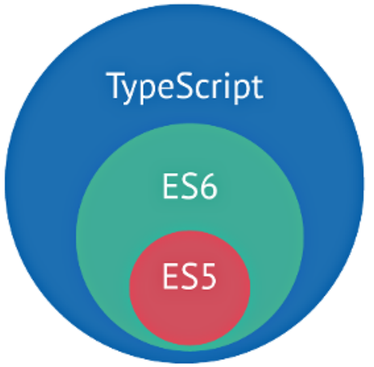
Extending the language

Go functional ?

- JavaScript has the basic paradigms for functional programming (lambas, closures, functions as data)
- It can easily be improved in that direction by adding static typing and better immutability

TypeScript && FlowType

- (Language)
  - is a superset of JS
  - compiles to JS (using its own compiler)

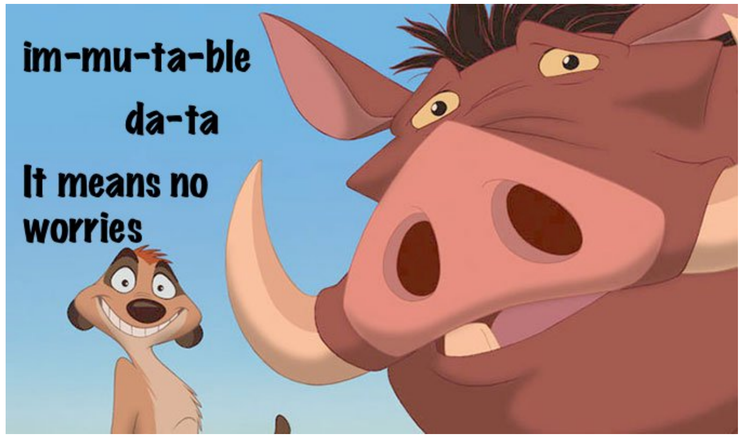


- (annotations)
  - Based on OCaml compilation chain
  - adds on existing JS
  - syntax and transformations plugins available on Babel

Advantages of adding static types to your JS

- Enforces conventions
- Most errors are caught at compilation time
- Less defensive code

- Provides various immutable data types, such as:
  - List
  - Map
  - Record
  - Set



Packaging

NPM

- Is a repository available on (anyone can publish)
- Is a tool shipped with NodeJS
- package.json
- node_modules

NPM

- Quantity of packages is plethoric
- Quality of packages is poor
- Is “slow”
- Does not work offline

Yarn: the new kid in town

- Backward compatible with NPM (through package.json)
- Fast (local cache, parallelism,…)
- Deterministic / Reliable (yarn.lock)

Bower

- Was thought as a front-end dependency manager
- Slowly vanishing from the landscape in favour of NPM/YARN

“The more dependencies you take on, the more points of failure you have”

– David Haney (@haneycodes)

Front End

Window and Document

- window represents the current opened window/tab
- document represents the current webpage and its tree

Main APIs

- document.getElementById(id)
- element.getElementsByTagName(name)
- document.createElement(name)
- document.querySelector(query)
- document.querySelectorAll(query)
- parentNode.appendChild(node)

Main APIs

- element.innerHTML
- element.style
- element.setAttribute
- element.addEventListener
- window.onload
- window.pageXOffset

Modularity

- es-imports are not yet implemented in browsers (will they, ever ?)
- Sometimes you can not rely on a bundler
- import your scripts in dependant order (or not ?)
- use the window object to namespace your application
- hint: **window.namespace = window.namespace ll {}**

Reflows and Repaints

- A great power comes with a great responsibility
- Even though optimised, the browser tends to feel like an artist and paint whatever it thinks has changed
- DocumentFragment

Network bottlenecks

- The worst pain in front-end performance
- Ship as less code as you can
- Cache as much as you can

Fetch API

- Comes as a replacement for old fashioned XHR
- Simple API
- Returns a Promise

```
fetch('https://api.hearthstonejson.com/v1/18336/frFR/cards.json')
  .then((response) => response.json())
  .then(obj => console.log(obj))
```

Readme:

Persons of Interest

Brendan Eich

@BrendanEich

Co-Founder of Brave browser. Co-Founder of Mozilla, Creator of Javascript

Douglas Crockford

Senior architect at Paypal, Trouble maker, Writer of Javascript the Good Parts

Kyle Simpson

@getify

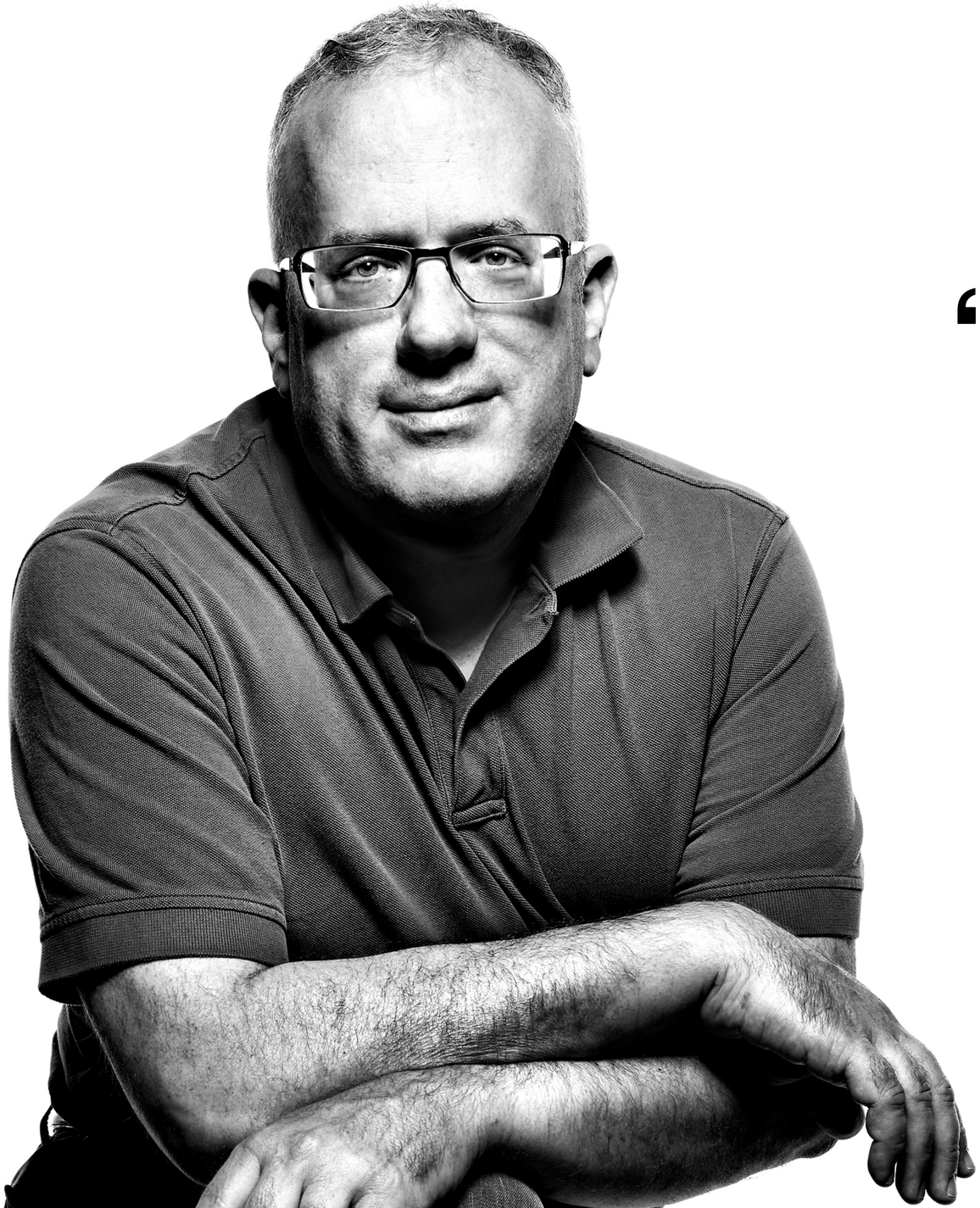
Software Engineer, Open Web Evangelist, Writer of You Don't Know Javascript

Reads

The Two Pillars of Javascript	Eric Eliott
Javascript: The Good Parts	Douglas Crockford
You Don’t Know Javascript (series)	Kyle Simpson
Mozilla Developer Network	

“You wanted a banana but what you got was a gorilla holding the banana and the entire jungle.”

Joe Armstrong about OOP



“Always bet on Javascript”

Brendan Eich