
# JavaScript - Lesson 1
# Language Basics

# Intervenant

- Quentin Pré
  <pre.quentin@gmail.com>

- MTI 2014


- Lead Front-End Engineer
  @ Adikteev / MotionLead


- I make digital ads for a living,

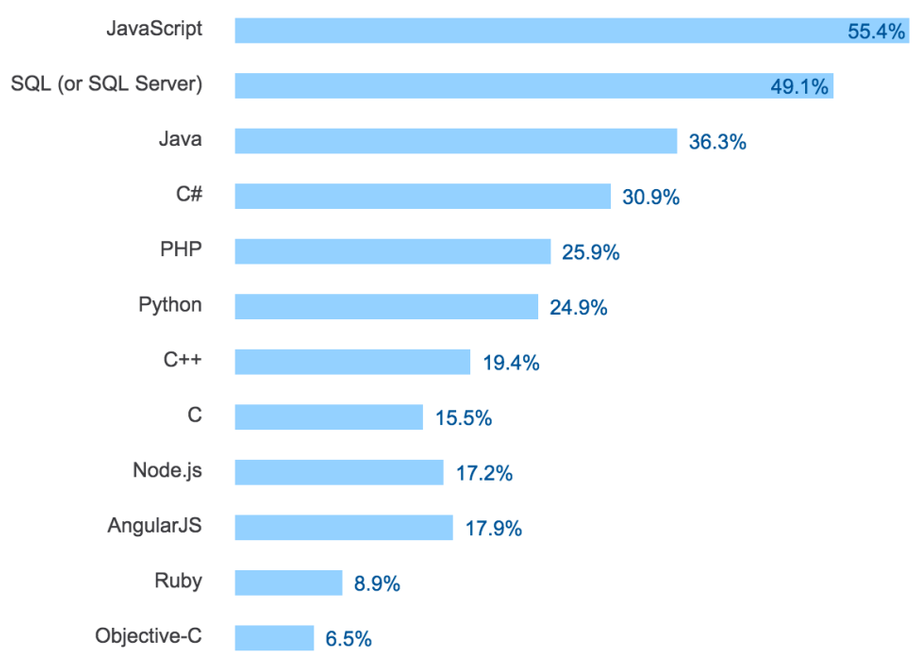

- I'm pretty sure there is a
  special place in hell for this.

# Who and Where?

# Well, everyone ?

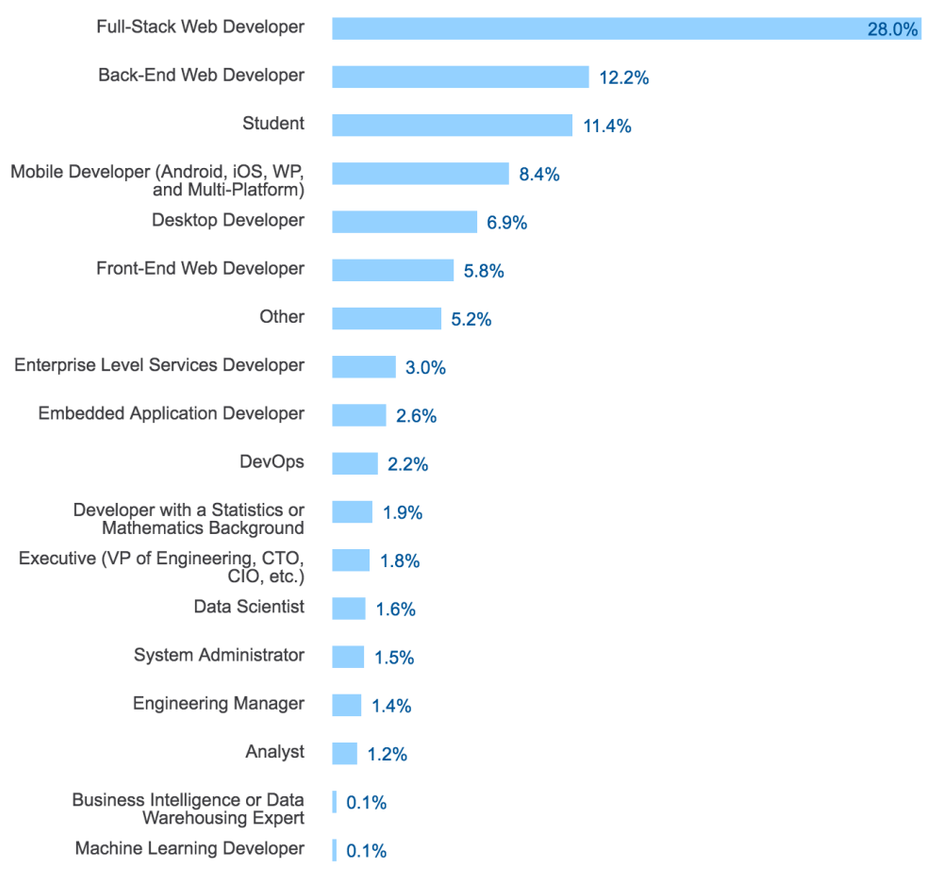
## I. Most Popular Technologies

2016 | 2015 | 2014 | 2013




*49,397 responses*

## II. Developer Occupations




*49,525 responses*

# Everywhere

- Complex browser apps
- Server (NodeJS)
- Desktop applications (Electron..)
- Mobile apps (React Native…)
- IoT

# History

# ECMAWHAT ?

# Origins

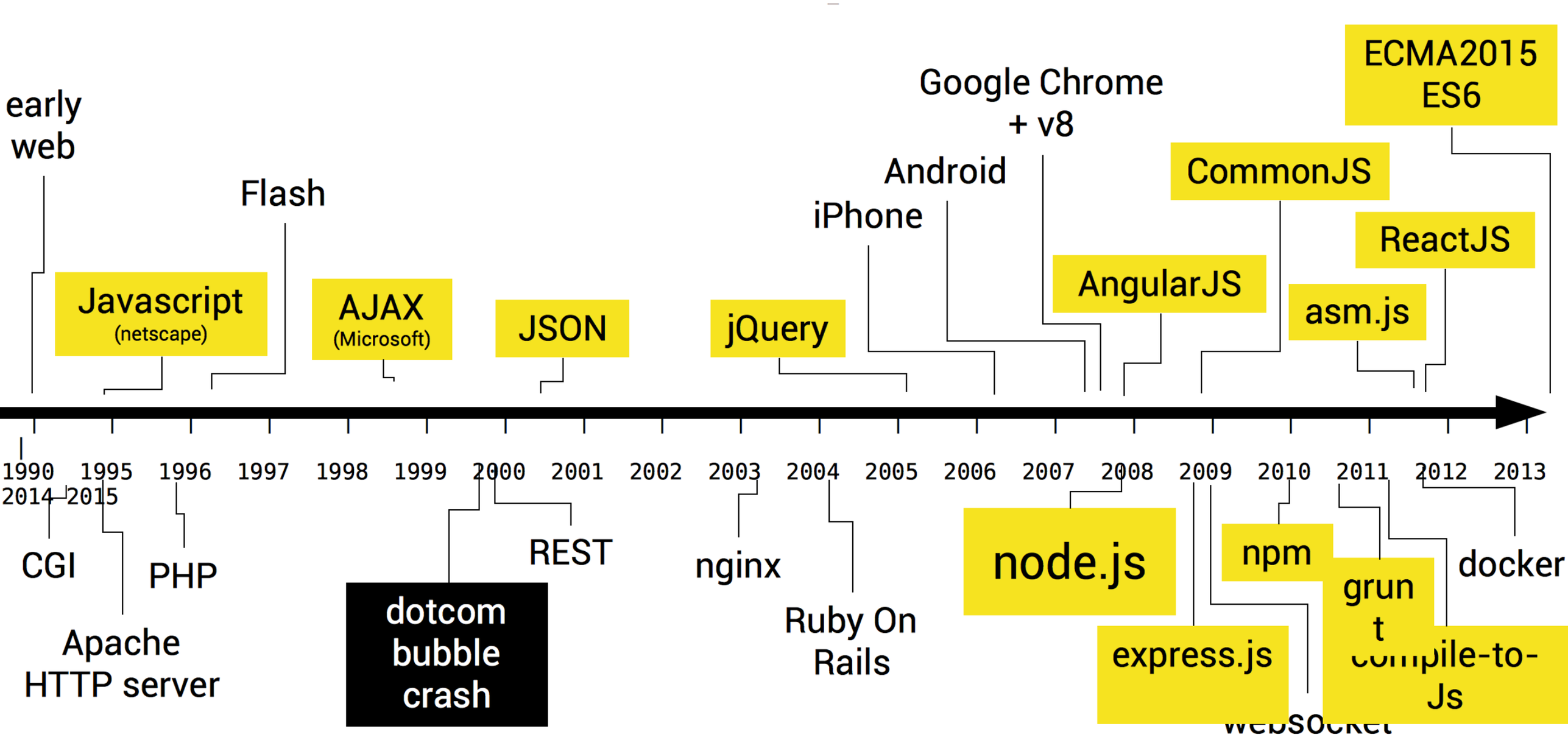
- Built by Brendan Eich in 1995 as part of Netscape
- Inspired by C and Java
- Built in 10 days
- Mocha -> LiveScript -> JavaScript
- First official spec: ECMA-262
- Standard name: ECMAScript

*“Always bet on Javascript”*
- Brendan Eich

# Early Days / pre-ES6

- 1998 - ECMAScript 2
- 1999 - ECMAScript 3
- 2000 - Starting development of ES4
- But Microsoft
- 2003 - Abort
- 2005 - ES4 spec is used to build AS3
- 2007 - Work on implementing AS3 in SpiderMonkey (Tamarin) is stalled
- 2007 - Yahoo, Google and MS release Javascript 3.1
- 2009 - 3.1 becomes ES5 (because they can), the first ubiquitous specification.

# Overview


early web
Flash
Javascript (netscape)
AJAX (Microsoft)
JSON
jQuery
iPhone
Android
Google Chrome + v8
AngularJS
CommonJS
ECMA2015 ES6
ReactJS
asm.js
1990
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
CGI
Apache HTTP server
PHP
dotcom bubble crash
REST
nginx
Ruby On Rails
node.js
npm
grun t
docker
express.js
compile-to-Js
WebSocket


# ES6 and beyond

- **ES6 === ES2015**
- **ES7 === ES2016**
- **ES Feature Proposals:**
  - **Stage 0 (strawman)** : someone from TC39 or a contributor thinks it is a good idea.
  - **Stage 1 (Proposal)** : TC-39 agrees it is a good idea.
  - **Stage 2 (Draft)** : First revision of the feature spec + 2 experimental implementations.
  - **Stage 3 (Candidate)** : Feature spec is finished + get feedback from implementations and users.
  - **Stage 4 (Finished)** : Feature spec is ready to be included in the next language spec

**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

# Useful links

- ES6 Specification
- ES6 Feature overview
- ES Proposals
- ES Compatibility Table

# Language Basics

# Generalities

- Javascript is a dynamic language
- It has both functional and object programming essential paradigms
- Runtimes use JIT (Just-In-Time) compilation to optimise (making it both interpreted and compiled)

Readme: The Two Pillars of JavaScript

Readme: JavaScript: The Good Parts

# Types && Operators

# Types

- Number
- String
- Boolean
- Object
    - Function
    - Array
    - Date
    - RegExp
- Symbol
- null / undefined

# Numbers

- **Double-precision 64-bits** values
- No **Integer** type
- **0.1 + 0.2**
- **Math** module provides functions (sin, cos…) and constants (such as PI)
- **parseInt(value: strineg, base: number) -> number**
  (careful though, **parseInt('toto', 10)** returns **NaN**)
- **NaN** for *Not A Number* -> avoid it.

Readme: WTF.js

# String

- Sequence of Unicode characters
- ```
  'hello'.charAt(0); // "h"
  'hello, world'.replace('hello', 'goodbye'); // "goodbye, world"
  'hello'.toUpperCase(); // "HELLO"
  ```
- See MDN for an extensive list of methods
  ```

# Boolean

- *false*, *0*, empty strings, *NaN*, *null* and *undefined* all become **false**

- All other value becomes **true**

- You usually do not need to convert a value to boolean javascript will silently convert values when it expects one.

# Object: basic

- Simple collection of name:value pairs
- create one using :
  **var obj = new Object()** or **var obj = {}**
- access properties with
  - dots: **obj.firstLevel.secondLevel**
  - square brackets: **obj[‘firstLevel’][‘secondLevel’]**
-

# Object: Array

- create one using :
  **var arr = new Array()** or **var arr = []**

- **map, filter, sort, reduce…** are your best friends when dealing with collections

  See MDN for more informations.

# Object: Functions

```
function add(a, b) { return a + b; }

const add = (a, b) => a + b; // implicit return

[1, 2, 3, 4].map(n => n * 2);

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

iife() // => ?
```

# Object: Functions

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

Student.prototype.say = function (something) {
  console.log(`${this.name} says ${something}`);
}
```

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

  say(something) {
    console.log(`${this.name} says ${something}`);
  }
}
```

# Operators

- Arithmetic: +, –, *, / and %
- assignment: =, +=, -=
- comparison:
- <, >, <= and >=
- == and !=

```
123 == '123'; // true
1 == true; // true
```

- === and !===

```
123 === '123'; // false
1 === true;    // false
```

# Operators: Spread (stage-2)

```
const arr = [1, 2, 3, 4];
const arr2 = [5, 6, 7, 8];

const arr3 = [...arr, ...arr2]; // => [1, 2, 3, 4, 5, 6, 7, 8]

function add(w, x, y, z) {
  return Array
    .from(arguments)
    .reduce((a, b) => a + b, 0);
}

add(...arr); // => 10
```

# Operators: Spread (stage-2)

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

const hasFooAndBar = {
  ...hasFoo,
  ...hasBar,
};

/*
  hasFooAndBar => { foo: 'foo', bar: 'bar' }
*/
```

# Scope && Variables

# Global vs Local

- Any variable declared outside of a function is in the global scope

- Any variable declared in a function can only be accessed from this it and its nested functions -> this is the **function scope**

# Function scope

- The canonical way of declaring a variable is the keyword `**var**`

```
var foo = 'bar'

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

function () {
  var toto = 'tata';

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

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

# Side effect: Closures

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

var add3 = makeAdder(3);
var add4 = makeAdder(4);

add3(3); // => 6
add4(1); // => 5
```

A closure is a combination of a function and a scope

# Function scope: the bad parts

```
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
```

# Block scope

- Therefore ES6 came with a new keyword for block-scoped declarations: **let**

```
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
```

- It also came with **const**, acting the same as **let** except that it is **read only**

```
const toto = 'foo';

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

# Well, almost read-only

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

foo.bar = 'foo';

// foo => { foo: 'bar', bar: 'foo' }
```

Readme: What const really stands for in ES6

# Sum up

# Destructuring

```
function getState() {
  return {
    name: 'square',
    width: 150,
    height: 150,
  };
}

function logWidth() {
  const { width } = getState();

  console.log(width);
}
```

# Hoisting

```
function fun() {
  foo(); // => TypeError "foo is not a function"
  bar(); // => "bar"
  var foo = function () { console.log('foo'); }
  function bar() { console.log('bar'); }
}

fun();

// is interpreted:

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

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

# Control Flow

# Nothing to declare

- If is still if
- while and do-while are still while and do-while
- for has brought its children:
  - for..of: iterates over values
  - for..in: iterates over keys (warning: it will include prototype keys)
  - Most of the times, use Array methods instead.

# Promise

- a Promise is an object wrapping a task and providing callbacks for control flow.

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

prom
  .then(() => console.log('yeeha !')); // => 'yeeha !', but after three secs

// you can register multiple resolvers
// on a single Promise
prom
  .then(() => console.log('yeehagain !')); // => 'yeehagain !', but after three secs
```

Readme: Callback hell

# 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

s ☐ 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% | 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


- babel-preset-env

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

- Now that most modern browsers support most ES6 features, only transform 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

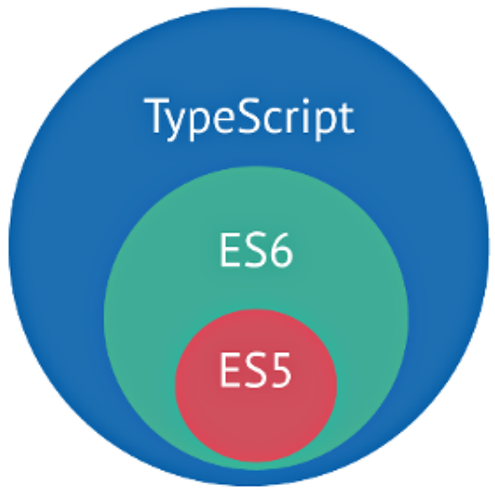
# Extending the language

# 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

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




- FlowType (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

# ImmutableJS

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

# Packaging

# NPM

- Is a repository available on npmjs.com (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: https://github.com/github/fetch

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

~ Joe Armstrong about OOP

“Questions ?”

– Every teacher at EPITA

Have a question later ?

<pre.quentin+jsct12018@gmail.com>

You can fetch the assets for the workshop here:

**https://github.com/qpre/epita-ing1-tp1**

prerequisites if you are not on the PIE:

- node 10.2
- yarn > 1.6