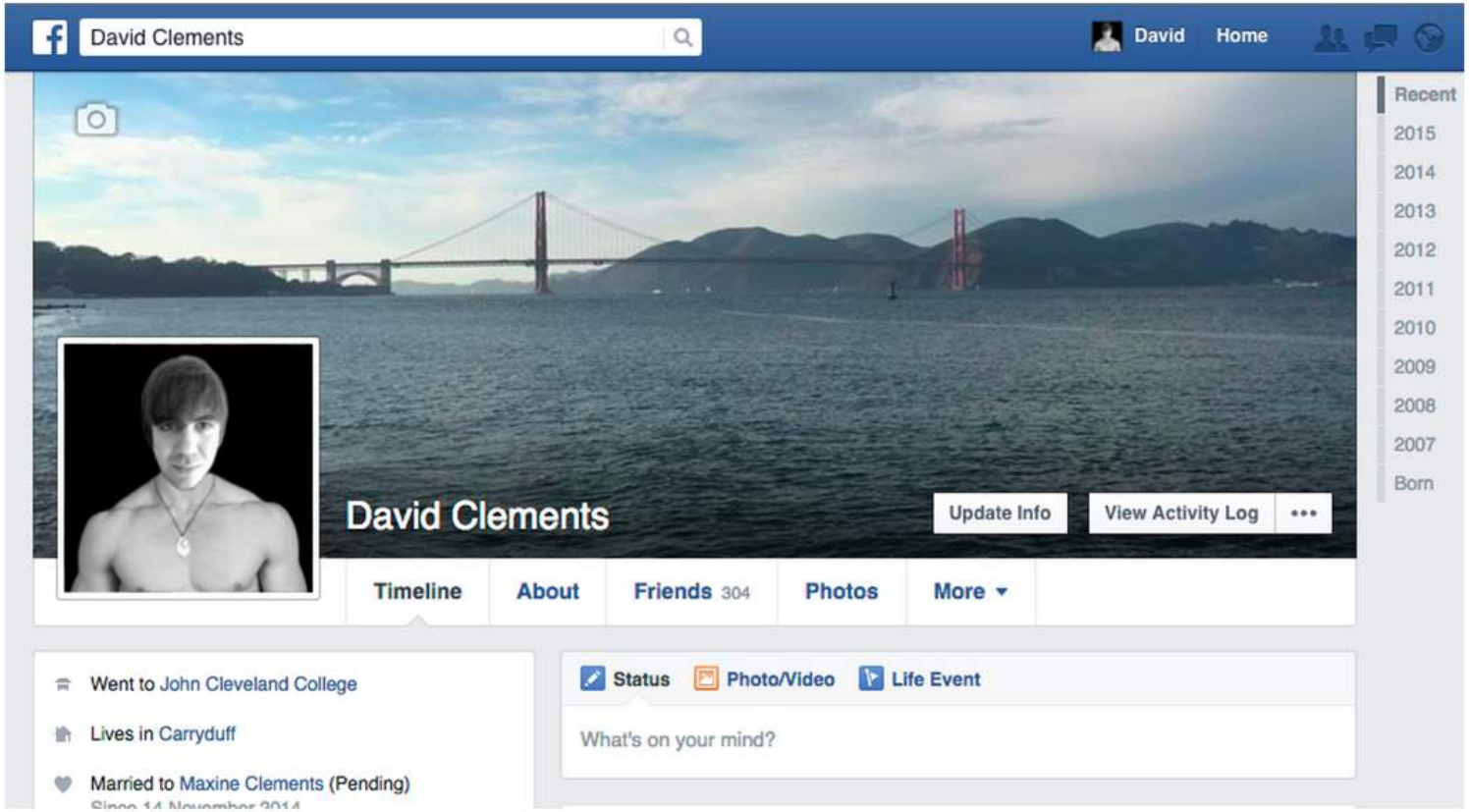
Growing up with JavaScript

History & Future of an Oddball

@davidmarkclem

Warning: Gratuitously Egotistical



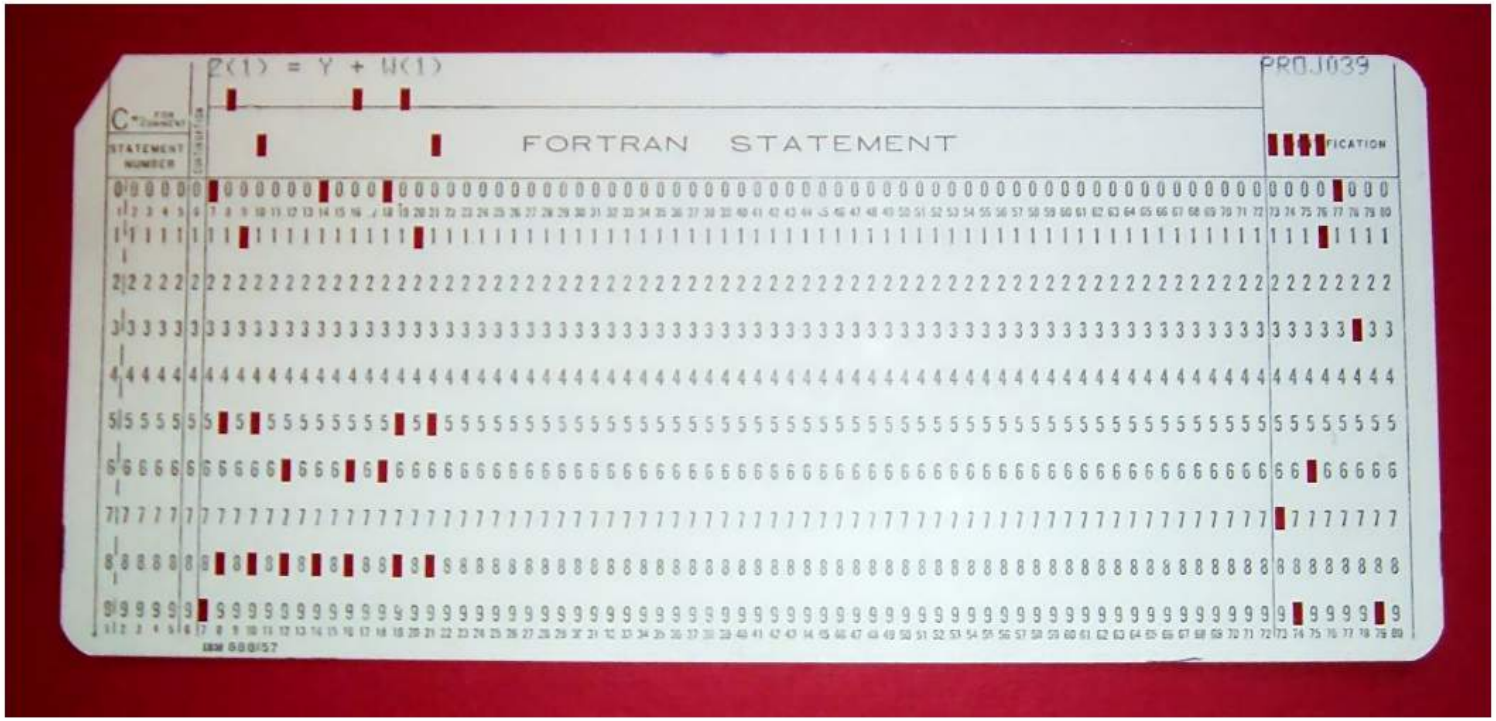
...more self-involved than a Facebook Profile

32 BD

32 BD

(1953)

FORTRAN



FORTRAN

```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPAY ERROR OUTPUT CODE 1 IN JOB CONTRO

      INTEGER A,B,C
      READ(5,501) A,B,C
  501 FORMAT(3I5)
      IF(A.EQ.0 .OR. B.EQ.0 .OR. C.EQ.0) STOP 1
      S = (A + B + C) / 2.0
      AREA = SQRT( S * (S - A) * (S - B) * (S - C) )
      WRITE(6,601) A,B,C,AREA
  601 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA=
      STOP
      END
```

27 BD

(1958)

ALGOL

```
FLOATING POINT ALGOL TEST'
BEGIN REAL A,B,C,D'

READ D'

FOR A:= 0.0 STEP D UNTIL 6.3 DO
BEGIN
  PRINT PUNCH(3),ff£L??'
  B := SIN(A)'
  C := COS(A)'
  PRINT PUNCH(3),SAMELINE,ALIGNED(1,6),A,B,C'
END'
END'
```

LISP

```
(LABEL,LENGTH, (LAMBDA, (Y), (COND, ((NULL, Y),
(QUOTE,0.0)), (T, (SUM, (LENGTH, (CDR, Y)),
(QUOTE, 1.0))))))
```

25 - 15 BD

(1960 - 1970)

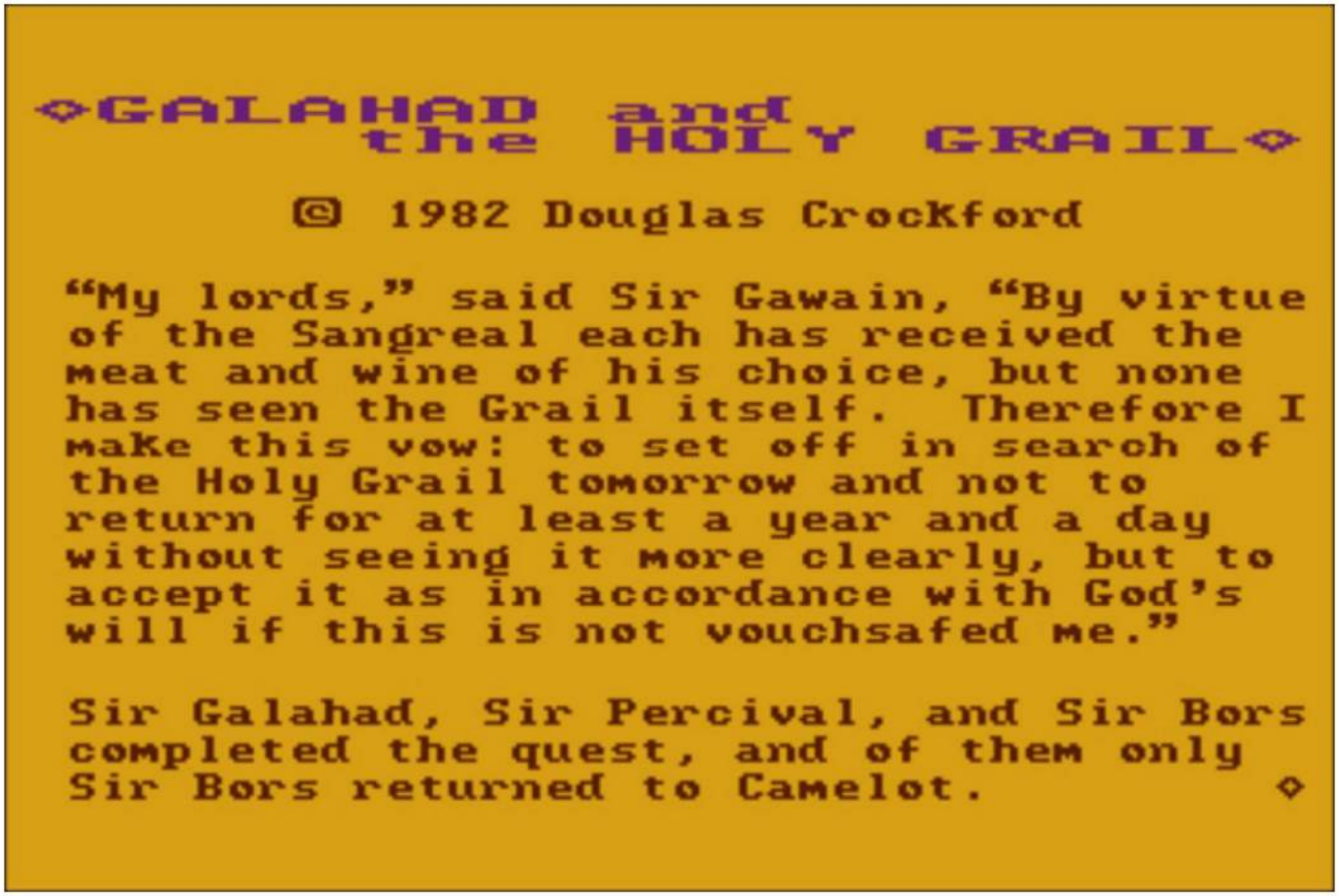
15 - 5 BD

(1970 - 1980)

3 BD

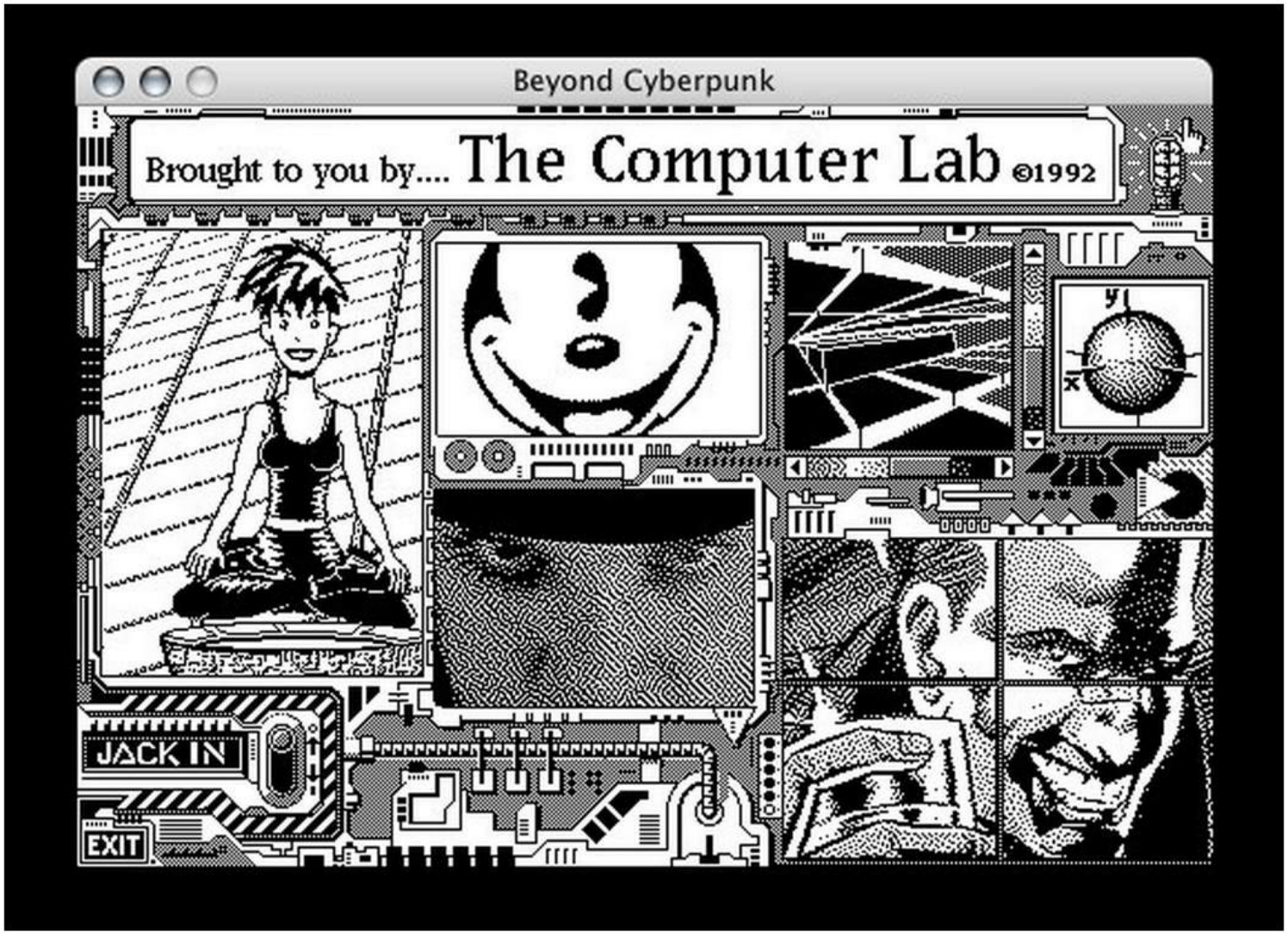
(1982)

Galahad and the Holy Grail



0 DB (1985)

Hypercard


Beyond Cyberpunk
Brought to you by.... The Computer Lab ©1992
JACK IN
EXIT

1986

Self

Hypertalk

```
on mouseDown
  answer file "Please select a text file to open."
  if it is empty then exit mouseDown
  put it into filePath
  if there is a file filePath then
    open file filePath
    read from file filePath until return
    put it into cd fld "some field"
    close file filePath
    set the textStyle of character 1 to 10 of card field "some field" to bold
  end if
end mouseDown
```

1987

CompuServe brings us GIF

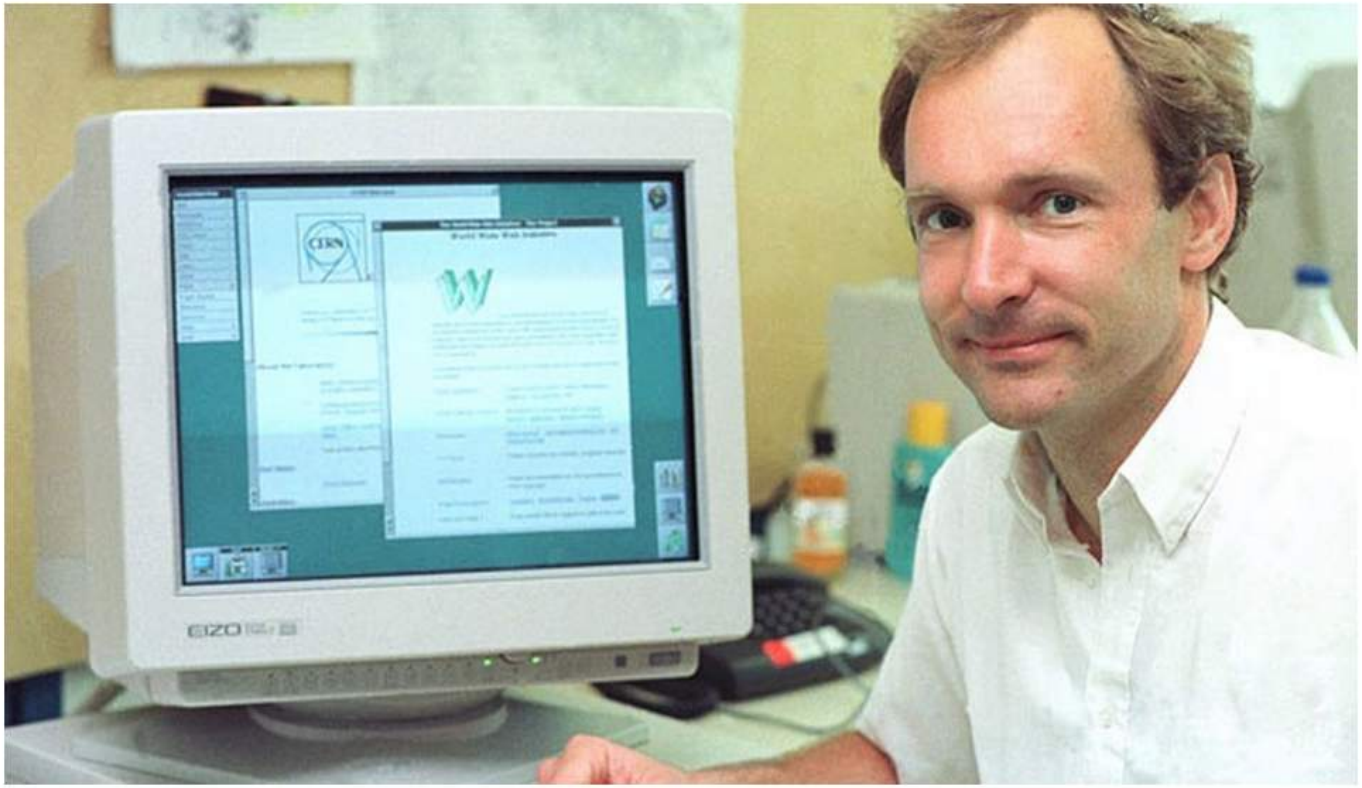
I was watching Rainbow

1989

And I was all like...

1990

The Architect



1991

My Christmas Present

My Babysitter

This is what he showed me...

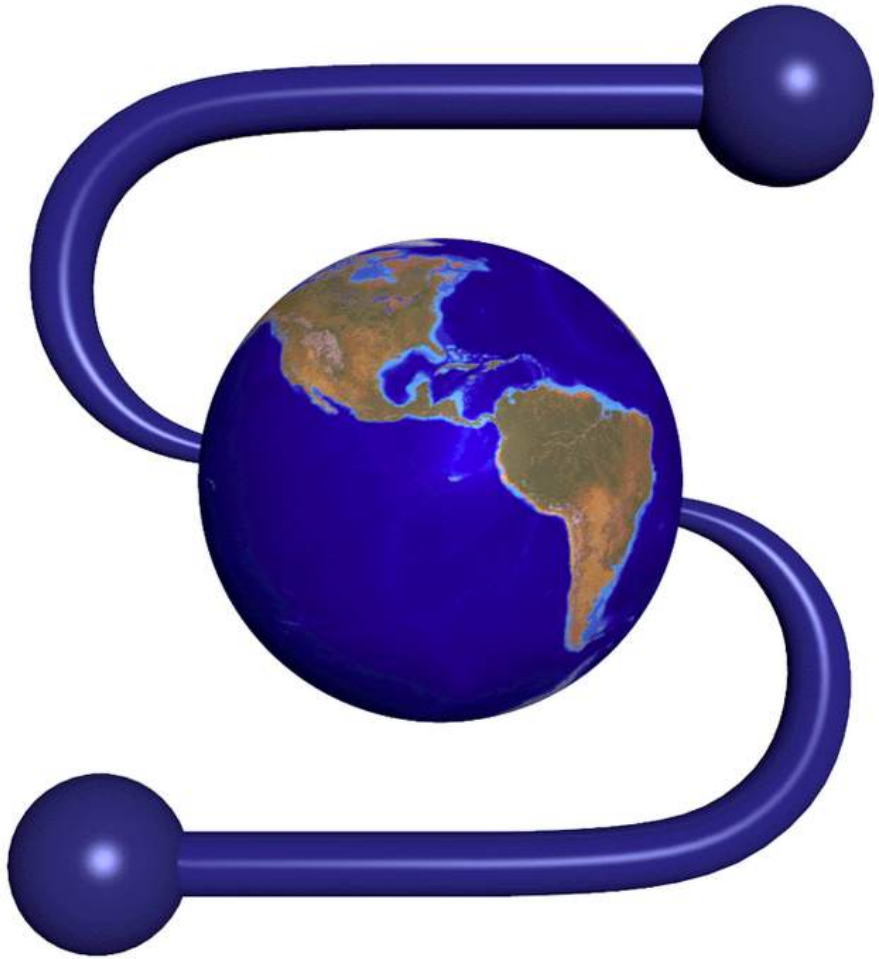
...I was astounded.

1992

Got me a mega drive

1993

Mosaic Released to the Public



Ataris!!!

Ataris!!!


RESET
POWER
L1
L2
START
SELECT
OPTION
F1
F2
F3
F4
HELP
BREAK
ATARI 1200XL
ESC
CLEAR
INSERT
DELETE BACK SPACE
CTRL
SET TAB
RETURN
CAPS
SHIFT
SHIFT

Ataris!!!

Ataris!!!

Ataris!!!

1994

Browsers Wars I

Netscape is born

1995

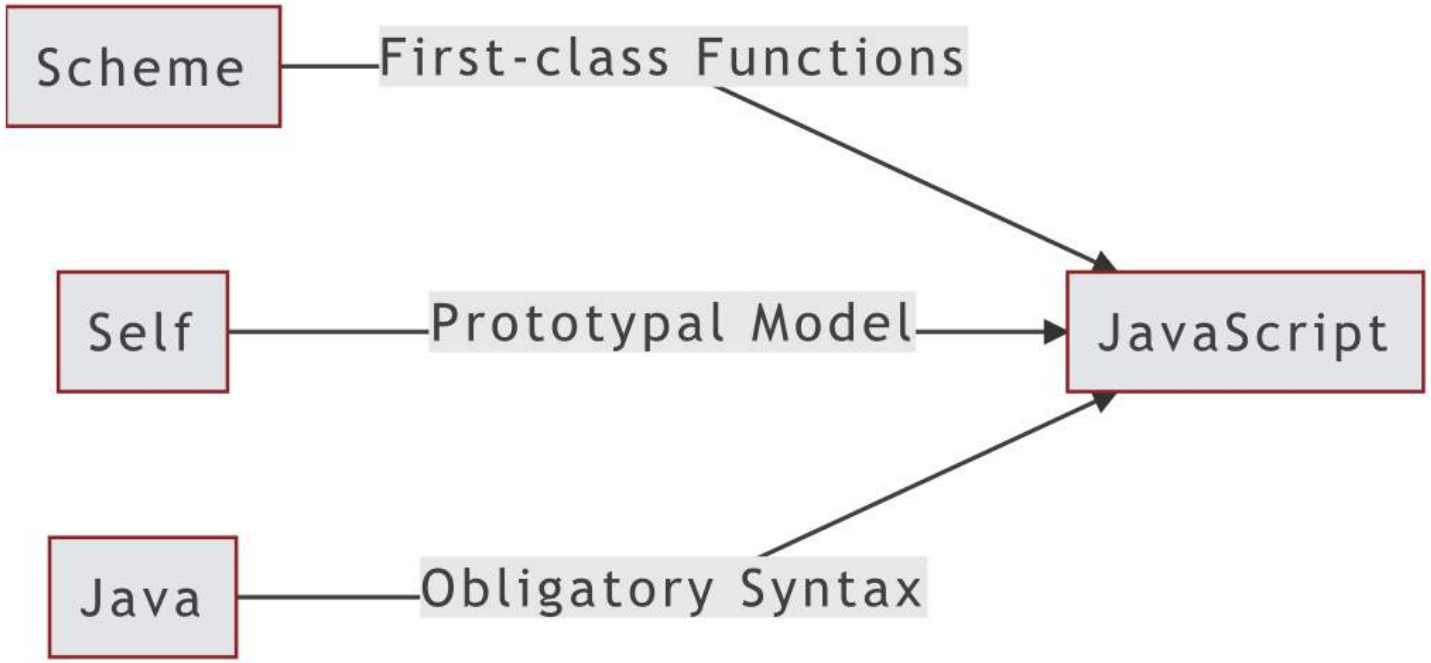
I ruined my Grandmas education

i386, win 3.1

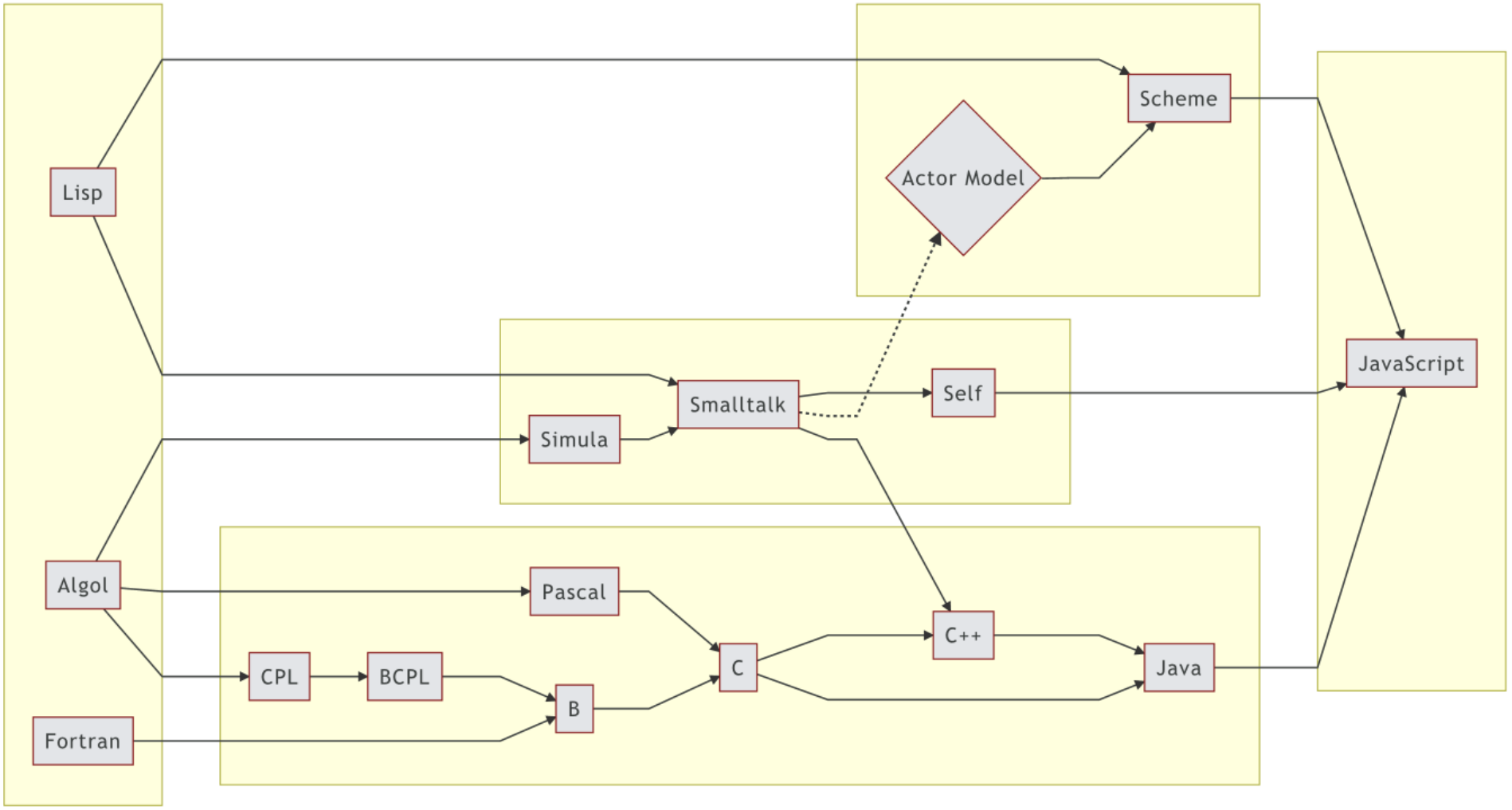
Internet Explorer is born

10 Days? np.

JavaScript Ingredients



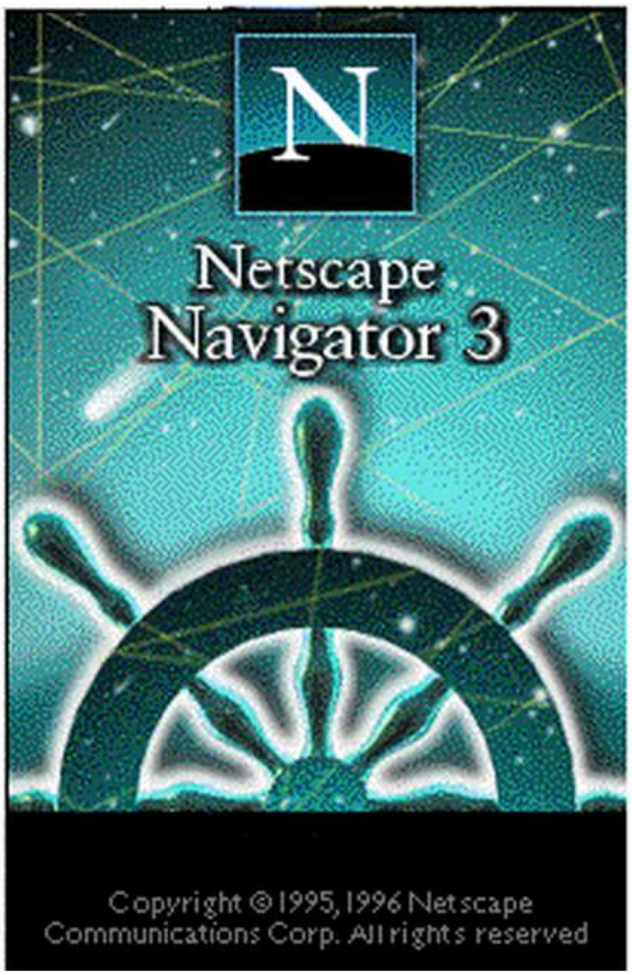
JavaScript Heritage



1996

Netscape 2.0, Netscape 3.0

JavaScript 1.0, JavaScript 1.1

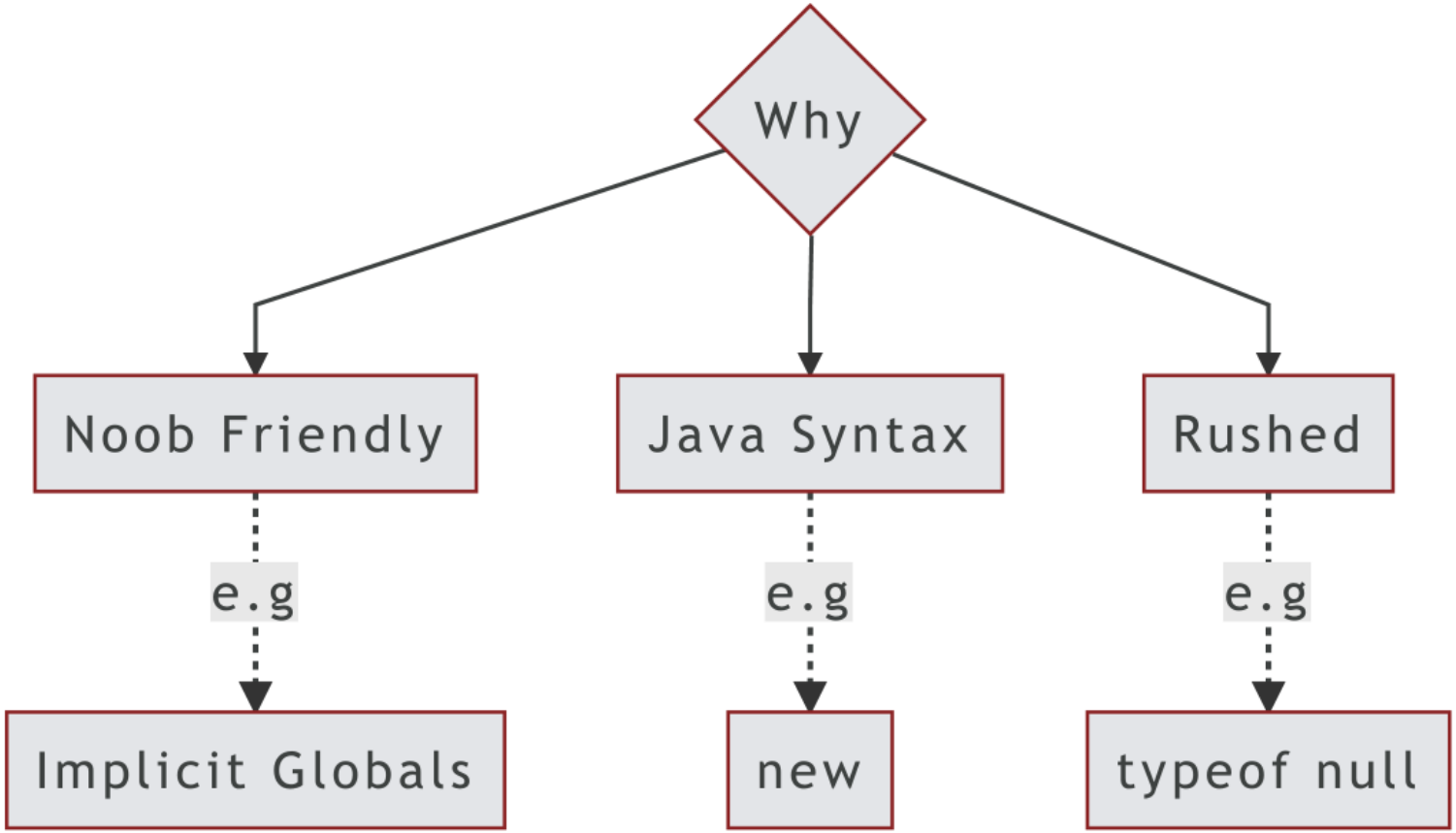


IE3 - JScript

JavaScript Sux

JavaScript Has Sucky Parts

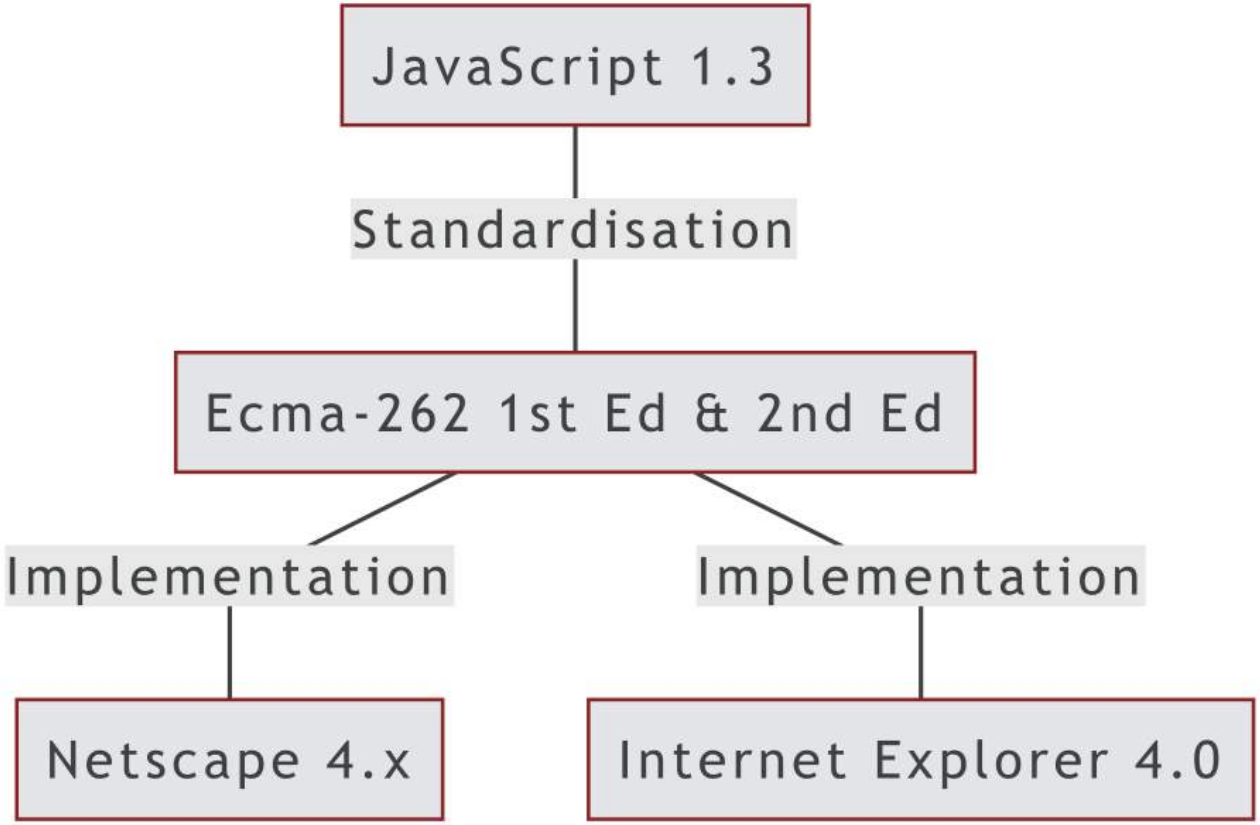
Sucky Parts


Why
Noob Friendly
Java Syntax
Rushed
e.g
e.g
e.g
Implicit Globals
new
typeof null

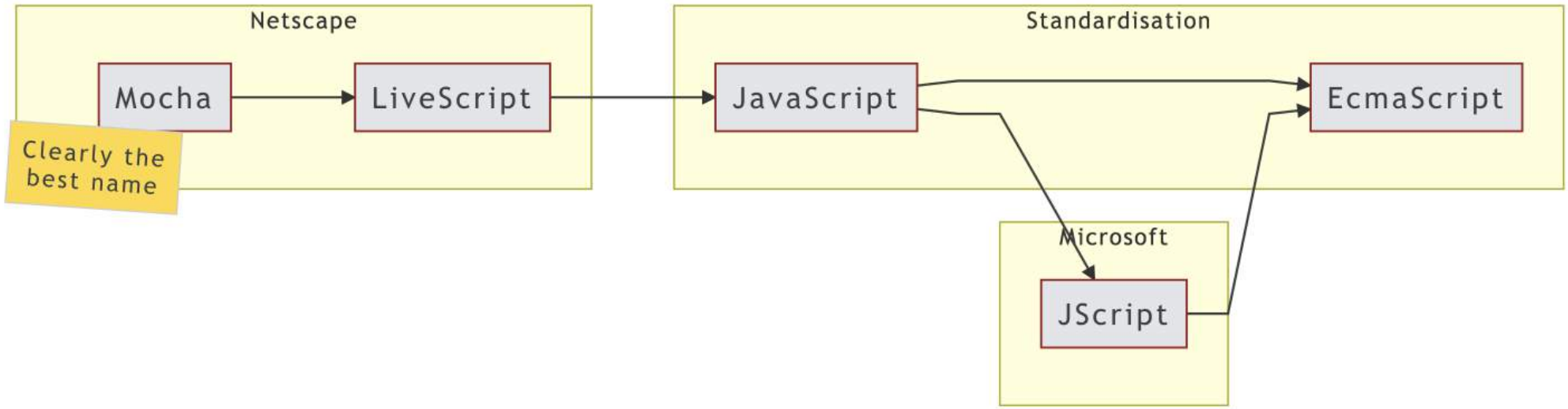
1997-1998

JavaScript Gets Standardised

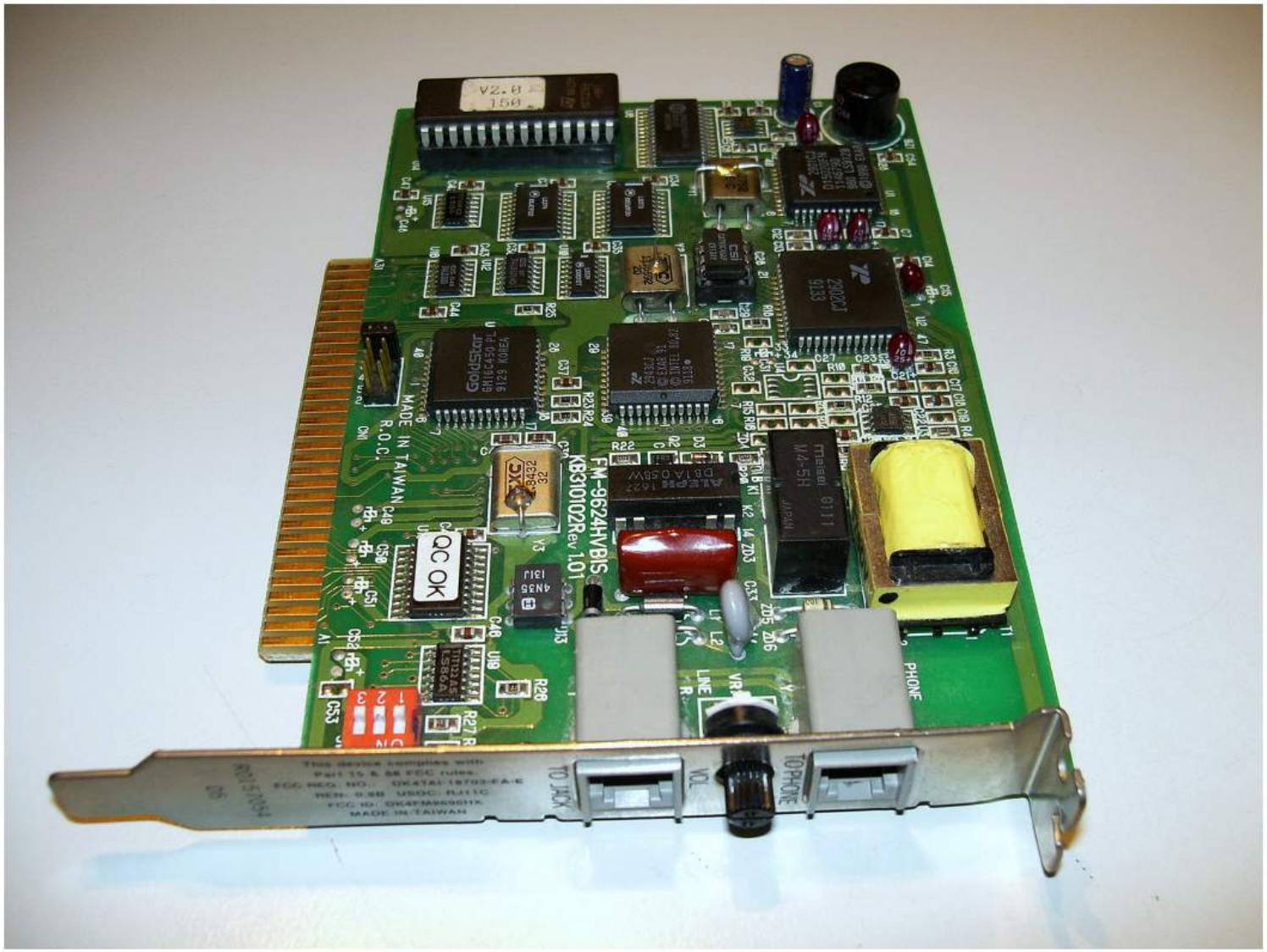
JavaScript Gets Standardised



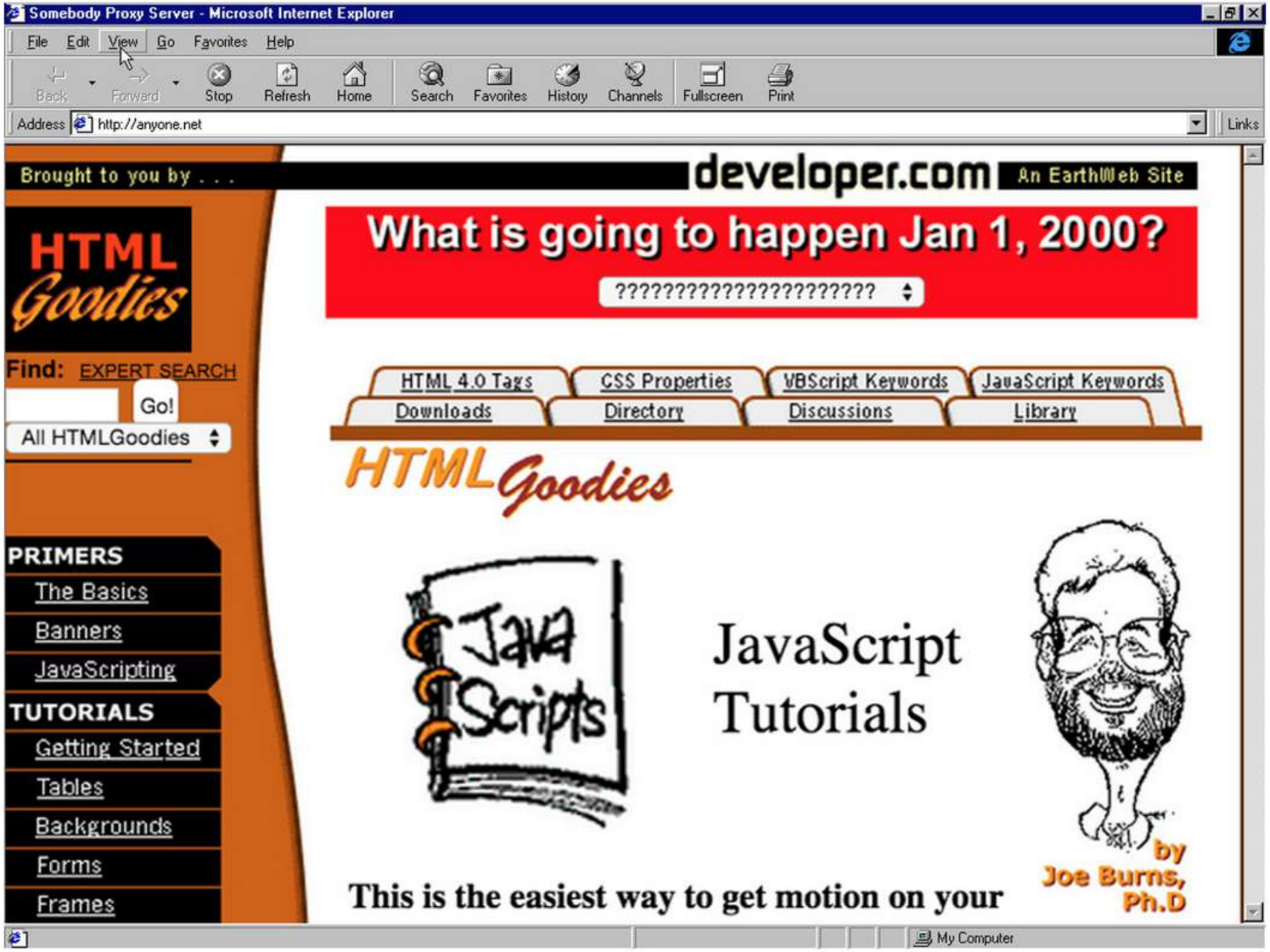
One of the two hard things...



I "rung up" up a £300 phone bill



Dialing up for this action:



1998

Netscape Open Sources Codebase, Calls it Mozilla

2000

IE 5.5 - EcmaScript 3rd Edition

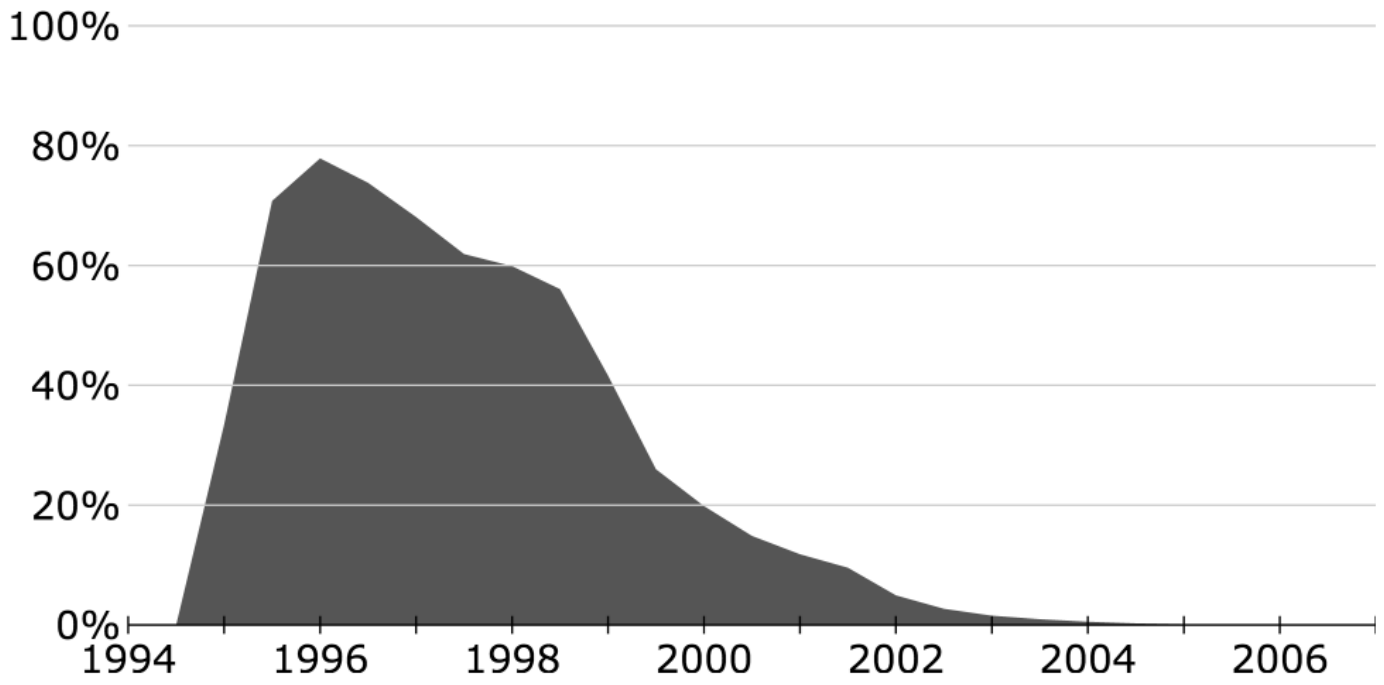
Netscape 6

It isn't good.

2001

IE 6 - EcmaScript 3rd Edition

Netscape Market Share: 9%



2002

This guy...

...launches JSON.org

Introducing JSON

العربية Български 中文 Český Dansk Nederlands English Esperanto Français Deutsch Ελληνικά עברית Magyar Indonesia
Italiano 日本 한국어 فارسی Polski Português Română Русский Српско-хрватски Slovenščina Español Svenska Türkçe Tiếng Việt

ECMA-404 The JSON Data Interchange Standard.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

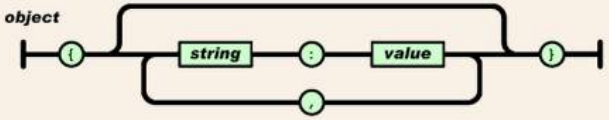
JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

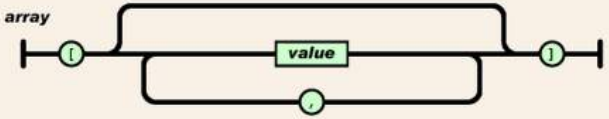
These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

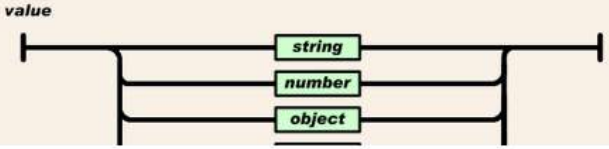
An *object* is an unordered set of name/value pairs. An object begins with `{` (left brace) and ends with `}` (right brace). Each name is followed by `:` (colon) and the name/value pairs are separated by `,` (comma).



An *array* is an ordered collection of values. An array begins with `[` (left bracket) and ends with `]` (right bracket). Values are separated by `,` (comma).



A *value* can be a *string* in double quotes, or a *number*, or `true` or `false` or `null`, or an *object* or an *array*. These structures can be nested.



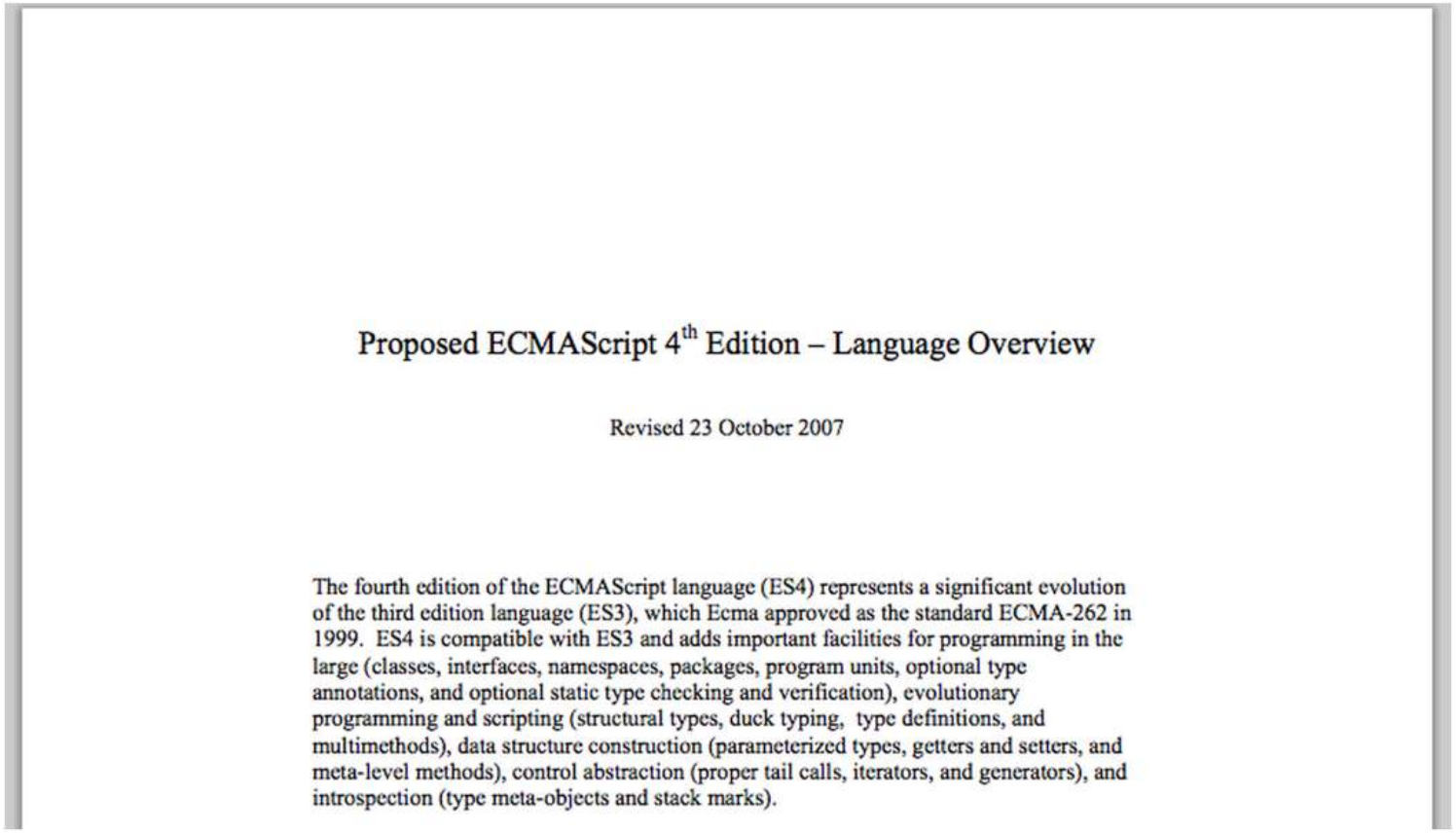
object
{}
{ *members* }
members
pair
pair , *members*
pair
string : *value*
array
[]
[*elements*]
elements
value
value , *elements*
value
string
number
object
array
`true`
`false`
`null`

---

string
""
" *chars* "
chars
char
char chars
char
any-Unicode-character-except-"-or-\-or-control-character
\"
\\
\/
\b
\f
\n
\r
\t
\u *four-hex-digits*
number
int

2007

EcmaScript 4 Whitepaper



Proposed ECMAScript 4th Edition – Language Overview

Revised 23 October 2007

The fourth edition of the ECMAScript language (ES4) represents a significant evolution of the third edition language (ES3), which Ecma approved as the standard ECMA-262 in 1999. ES4 is compatible with ES3 and adds important facilities for programming in the large (classes, interfaces, namespaces, packages, program units, optional type annotations, and optional static type checking and verification), evolutionary programming and scripting (structural types, duck typing, type definitions, and multimethods), data structure construction (parameterized types, getters and setters, and meta-level methods), control abstraction (proper tail calls, iterators, and generators), and introspection (type meta-objects and stack marks).

It was not well received...

2008

This guy...

Releases this book

Google launch Chrome, and with it the v8 JavaScript engine

TC39 Moves in new direction: EcmaScript 3.1

These guys were catalysts...

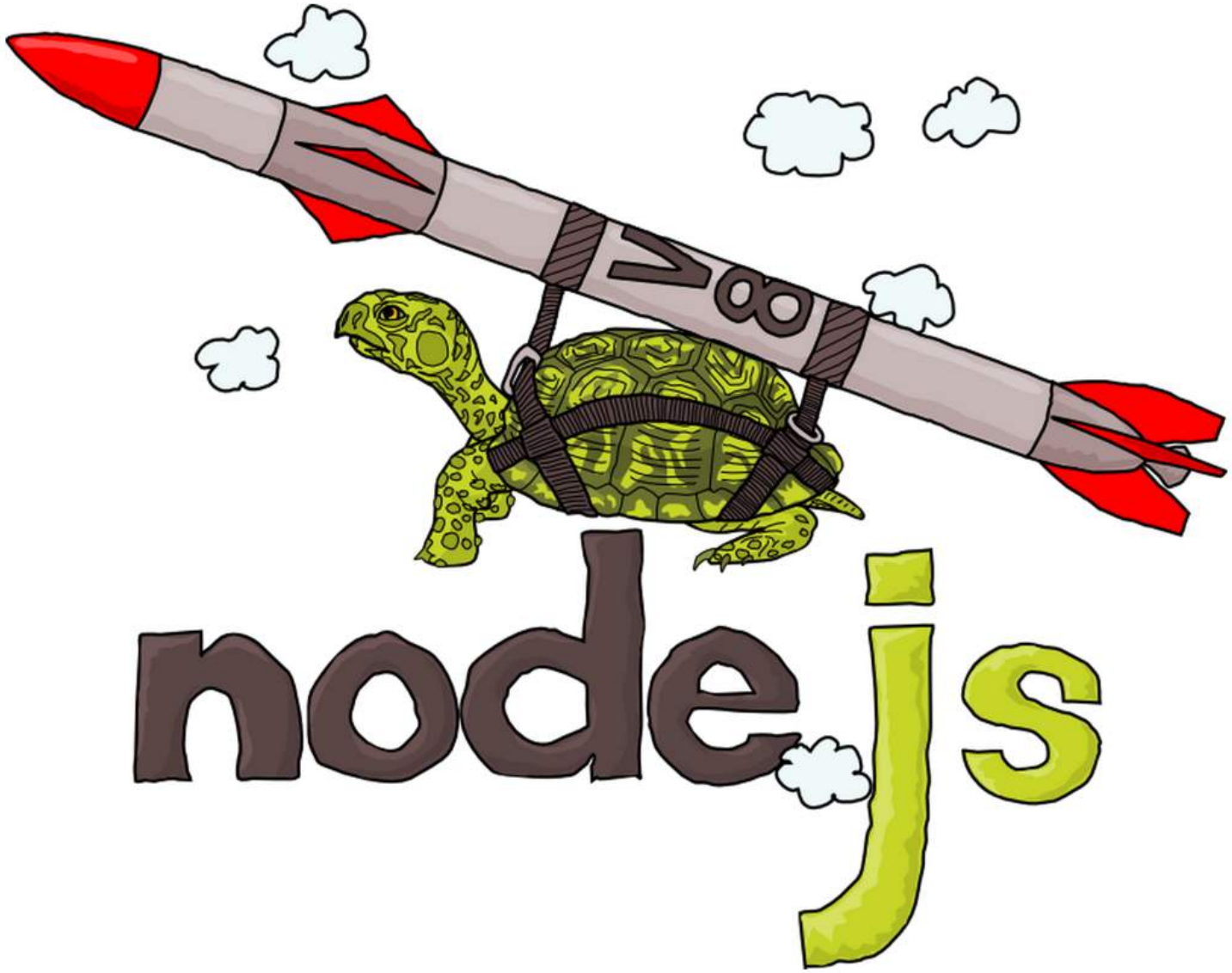
"ES.next" (ES6) also announced as syntactic extensions specification

2009

EcmaScript 3.1 becomes EcmaScript 5

Final TC39 Approved Draft Released

Ryan Dahl creates Node.js



Firefox 3.5 comes with the SpiderMonkey JS engine

Firefox® 3.5

Chris Williams organises the first JSConf

Chris Williams writes node-serialport, triggering...

The rise of JS Robots

2011

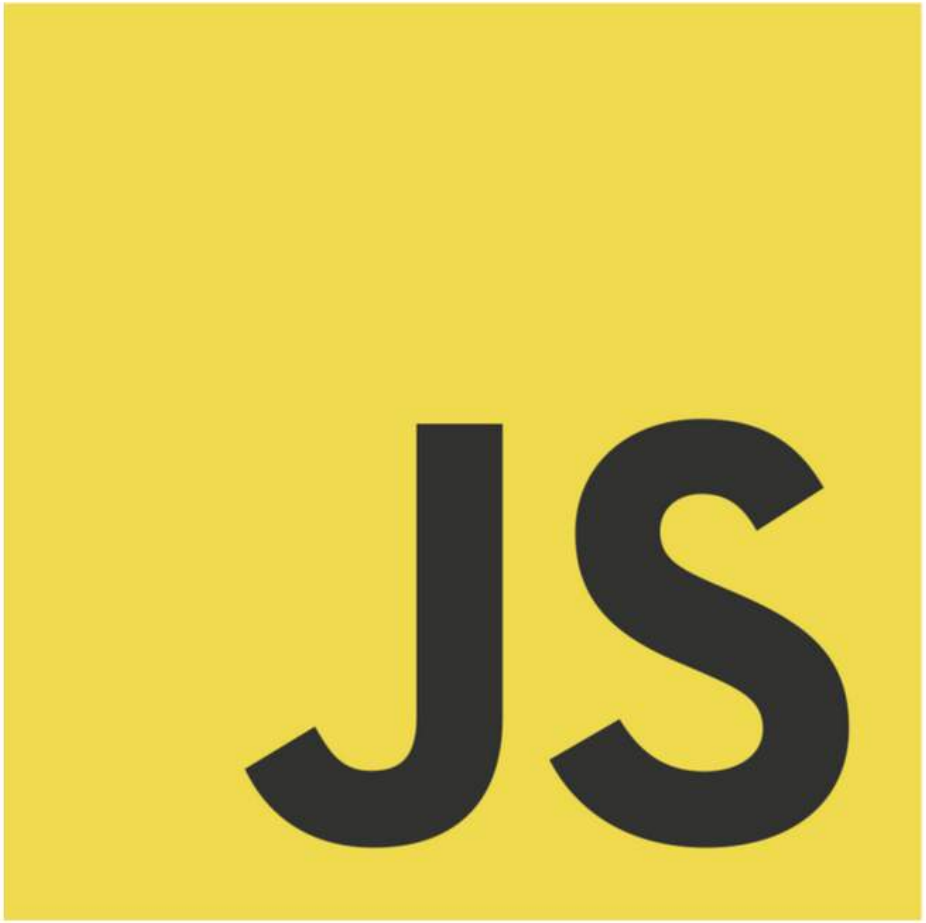
EcmaScript 5.1

EcmaScript 5

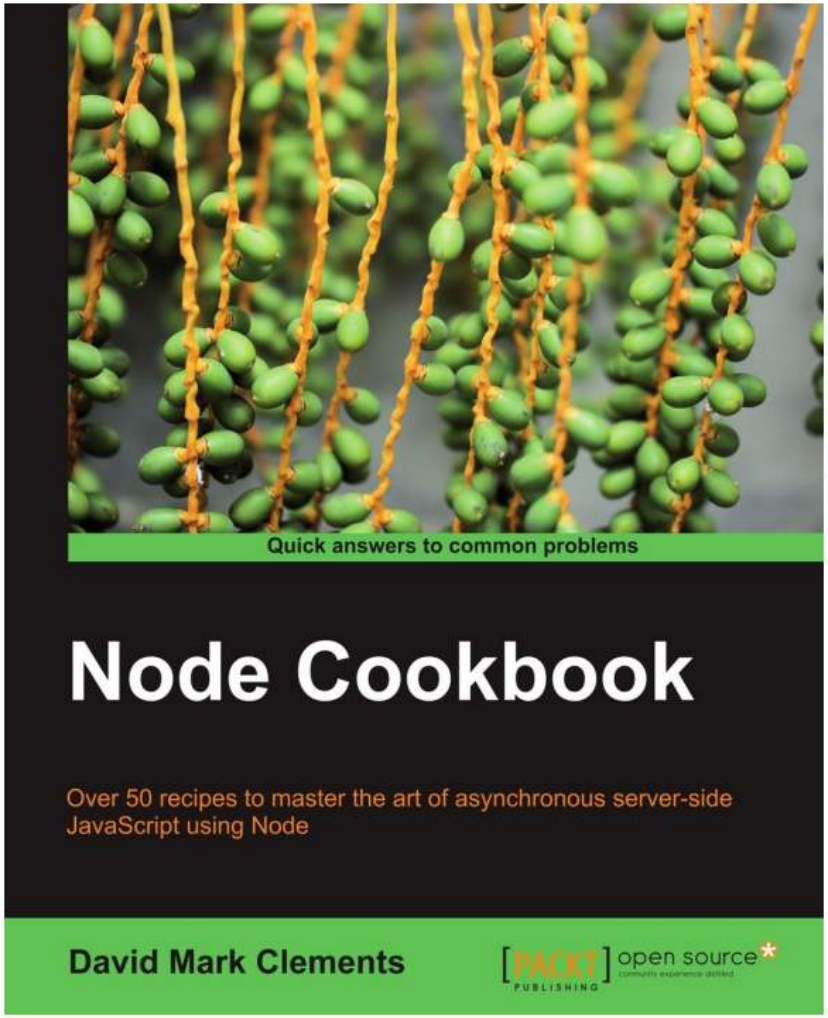
- Strict mode
- Array
  - forEach, map, reduce, filter
- Object
  - keys, create, defineProperties, freeze
- String
  - trim
- Function
  - bind
- JSON

2012

Chris Williams creates unofficial JS logo



Shameless Plug: A common idiom



2013

Node reaches 0.8

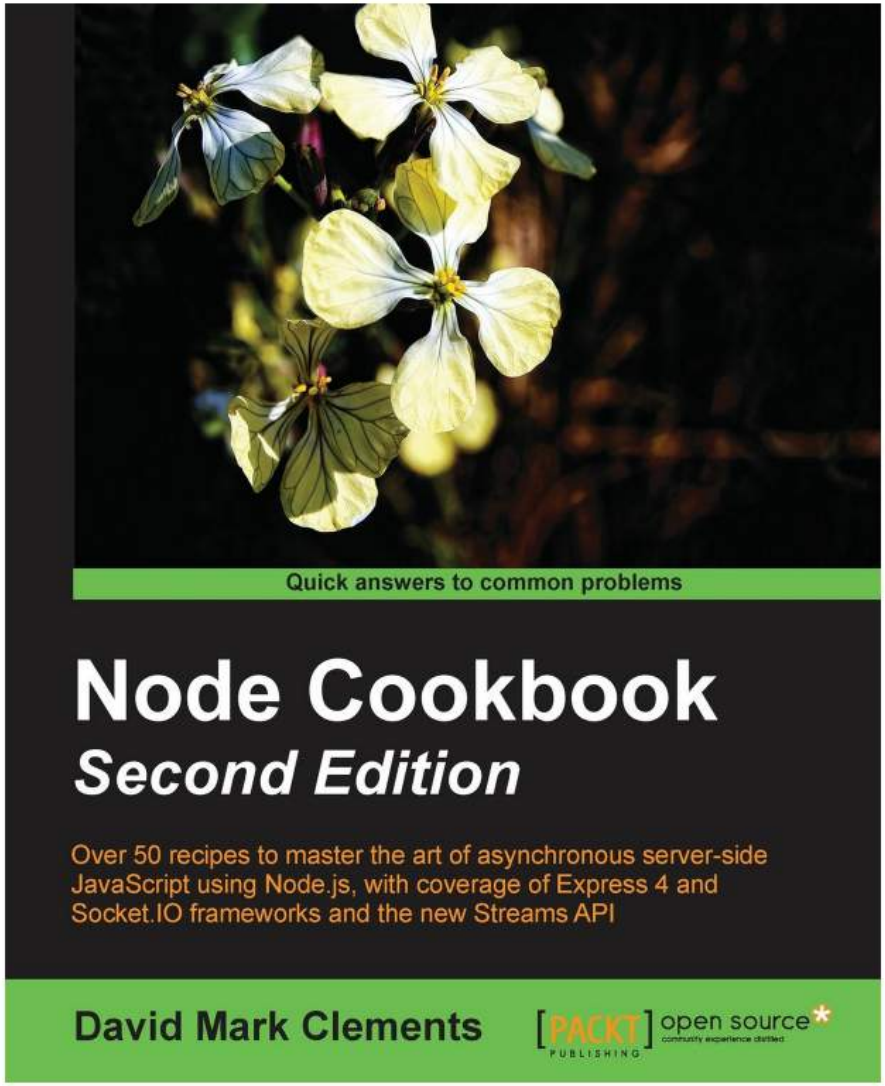
Early-adopters in Enterprise
begin to use it in production

2014

Node reaches 0.10

Good, stable all round improvements, easy migration

Node Cookbook Update

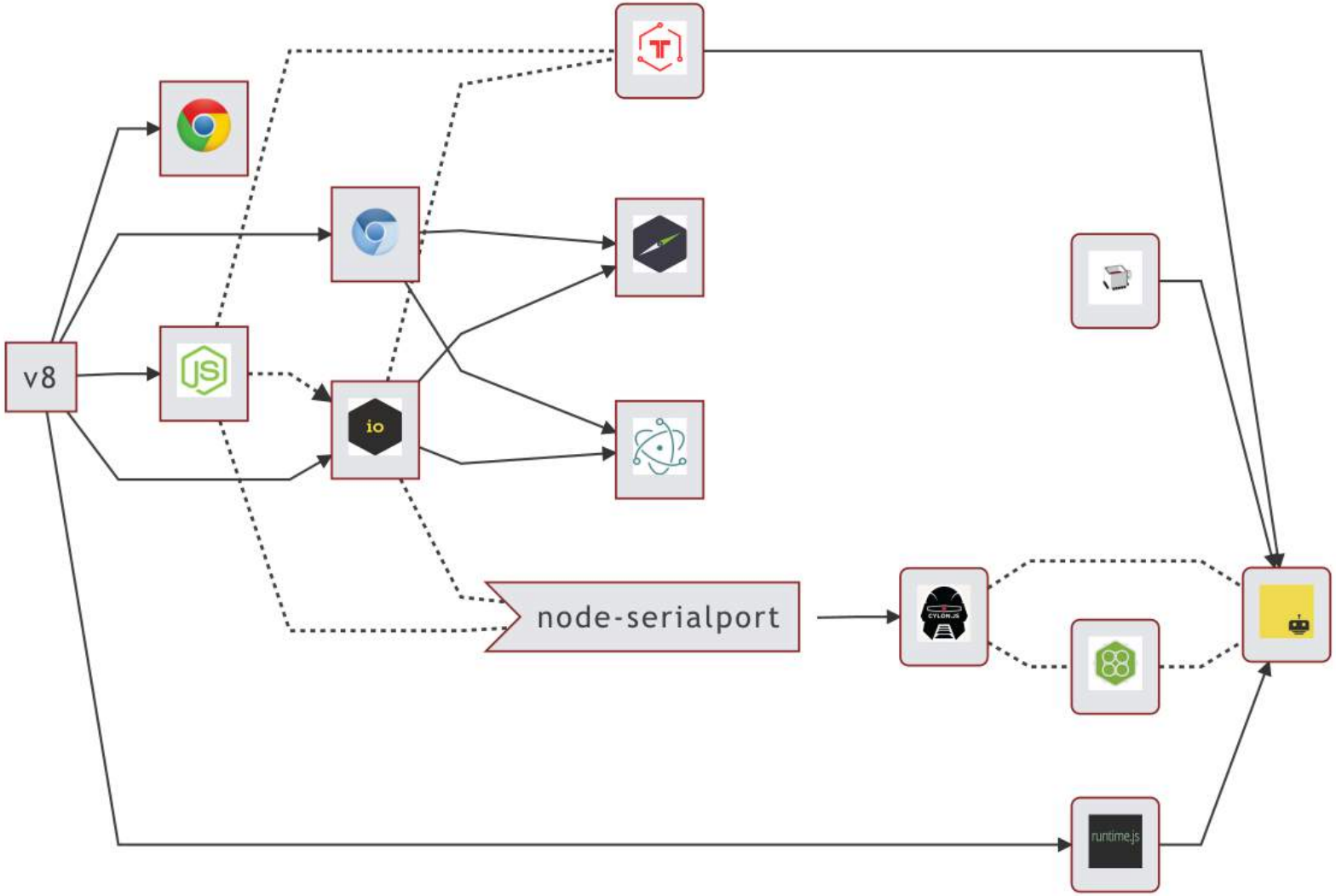


io.js fork

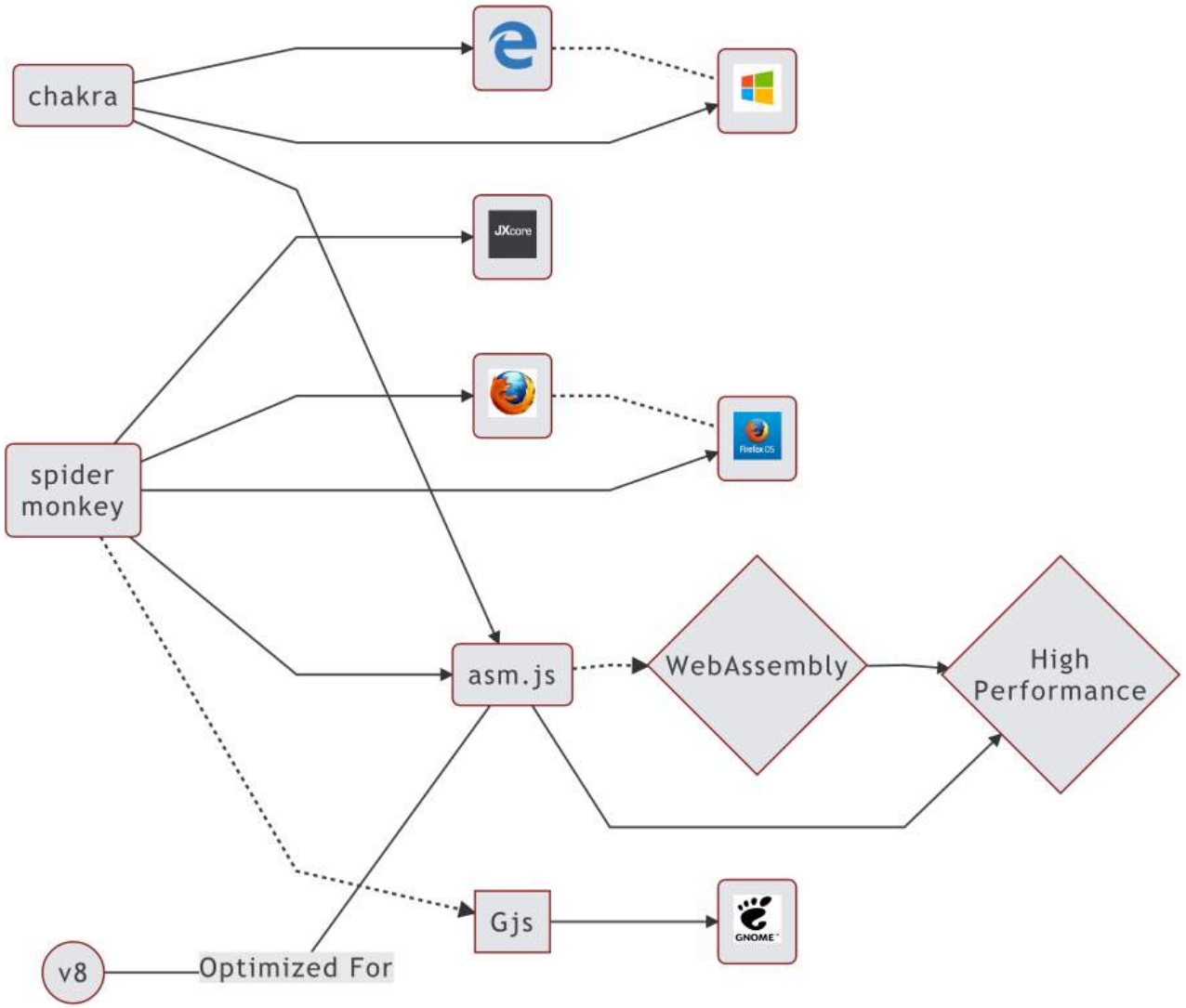
Developer discontent leads to fork

2015

JS Overview



JS Overview


chakra
JXcore
spider
monkey
asm.js
WebAssembly
High
Performance
Gjs
v8
Optimized For

Node reaches 0.12

Projects stewards motivated by fork to release
Best to stick with 0.10 while 0.12 matures a little

Creation of Node Foundation

Agreement between io.js and node project to merge

ES6 renamed to ES2015

ES7 === ES2016 ... ?

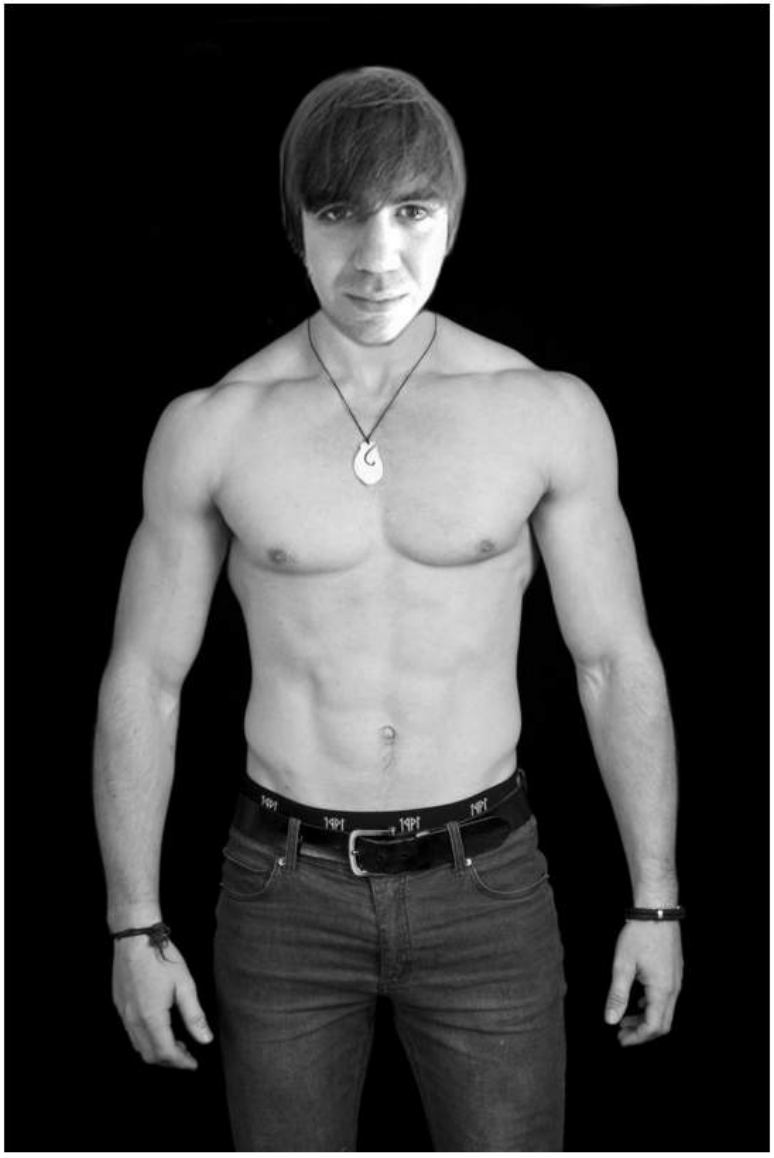
ES2015 Approved

June 17th 2015

Future

Soylent

Future Me



EcmaScript 6

- Lots of implementing to be done yet
- Chrome, Firefox, io.js - latest support
- Transpilers
- let, const (block scoping)
- Promises, Generators
- Classes (sugar), Modules
- Spread, rest, destructuring
- Lambdas, and lots more

EcmaScript 7 (2016?)

- Async/Await
- Guards
  - Enforcing arbitrary constraints
  - Typing Primitives
- Event loop concurrency
  - Sharing state between event-loops
  - (instead of workers/multiple processes)

Future future

EcmaScript 8 (2017?)

- Macros
- SIMD
  - Single instruction, multiple data
  - Parallel array processing

Future Future Me

Thank you.

Thank you.

@davidmarkclem