

Details matter when evolving the world's most widely used programming language



# **TEN** “SEVEN DAYS IN MAY”

**1995: Brendan Eich creates  
JavaScript**

THE JOHN C. MEEHLE PRODUCTION  
PRODUCED BY EDWARD LEWIS • DIRECTED BY JOHN FRANKENHEIMER  
A J.M.A. RELEASE

# “SEVEN DAYS IN MAY”

- May 1995, Created in ten days by Brendan Eich at Netscape: “Mocha”
- September 1995, shipped in beta of Netscape Navigator 2.0: “LiveScript”
- December 1995, Netscape 2.0b3: “JavaScript”

**1995: Brendan Eich creates  
JavaScript**



# Brendan Eich Interview, Early 1996

“So we saw a need for an interpreted-from-source, dynamically typed language with which one could orchestrate the interactions among HTML form elements and links, Java applets, plug-ins, and other components.”

“...make pages a little smarter and more live -- for instance, make a click on a link load a different URL depending on the time of day”

“The audience for this language, we hoped, would consist of HTML authors who had some programming experience”

“I'd like to see it remain small, but become ubiquitous on the web as the favored way of gluing HTML elements and actions on them together with Java applets and other components.”

# Things JavaScript 1.0 didn't have

object literals

function expression

most string methods, array methods, etc.

regular expressions

→ try/catch exception handling ←

```
> [1] + [2] - [3]  
9
```



# “SEVEN DAYS IN MAY”

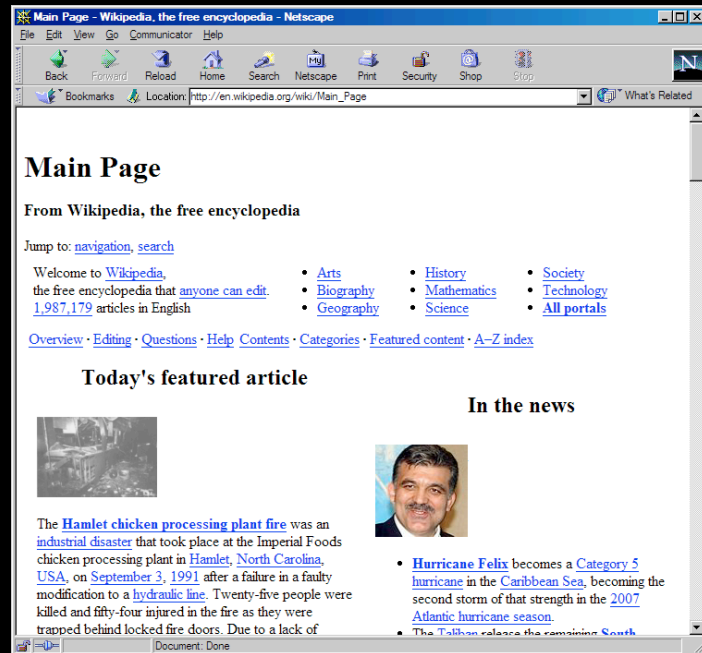
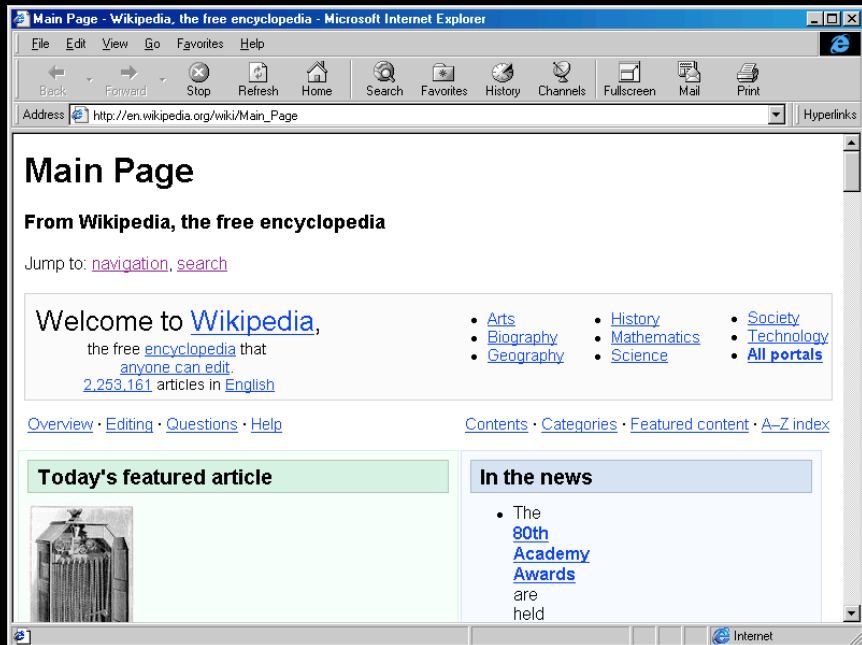
- May 1995, Created in ten days by Brendan Eich at Netscape: “Mocha”
- September 1995, shipped in beta of Netscape Navigator 2.0: “LiveScript”
- December 1995, Netscape 2.0b3: “JavaScript”
- **August 1996, JavaScript cloned in Microsoft IE 3.0: “JScript”**

THE JOHN FRANKENHEIMER-JOEL PRODUCTION  
PRODUCED BY EDWARD LEWIS • DIRECTED BY JOHN FRANKENHEIMER  
A PARAMOUNT RELEASE

# Browser Interoperability

Multiple independently created client browsers

All expected to meaningfully render the same content and work with all web applications



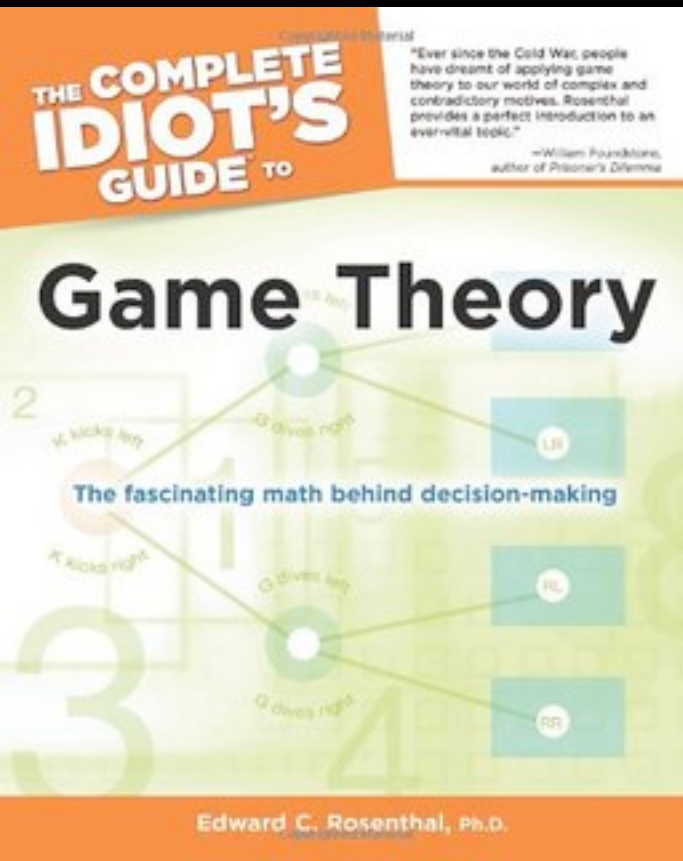




# Crockford's observation

On the web, the end-user chooses the deployment compiler.

# Browser Game Theory



New entrants must conform

Breaking changes (fixes) will drive away users

Innovation is wasteful, if only available in one browser

“First browser to try something new may lose market share, which will force it to go back to the bad old ways”



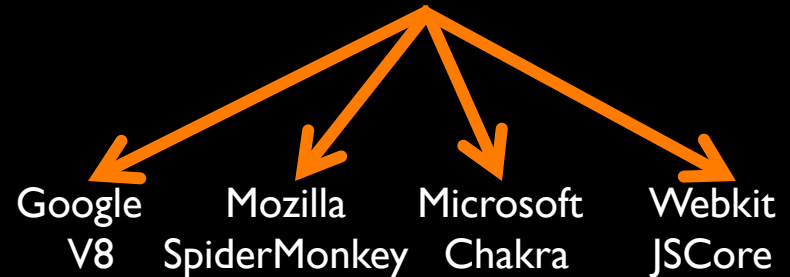
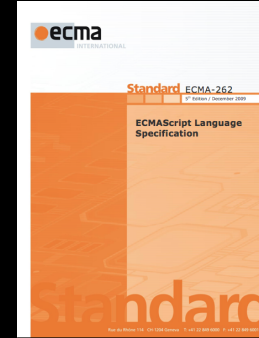
# “TEN DAYS IN MAY”

- May 1995, Created in ten days by Brendan Eich at Netscape: “Mocha”
- September 1995, shipped in beta of Netscape Navigator 2.0: “LiveScript”
- December 1995, Netscape 2.0b3: “JavaScript”
- August 1996, JavaScript cloned in Microsoft IE 3.0: “JScript”
- 1996-1997, Standardization ECMA-262 Ed. 1: “ECMAScript” aka ES1
- 1999, ES3 – modern JS baseline

1997: It's Time for a Standard

# What is ECMAScript?

- **ECMAScript** is the name of the international standard that defines the JavaScript programming language
- Developed by Technical Committee 39 (**TC-39**) of **Ecma International**
- Issued as document **ECMA-262**
- Not part of **W3C**



JavaScript Implementations

# Details really matter in an interoperability specification

A detailed and highly prescriptive algorithmic specification

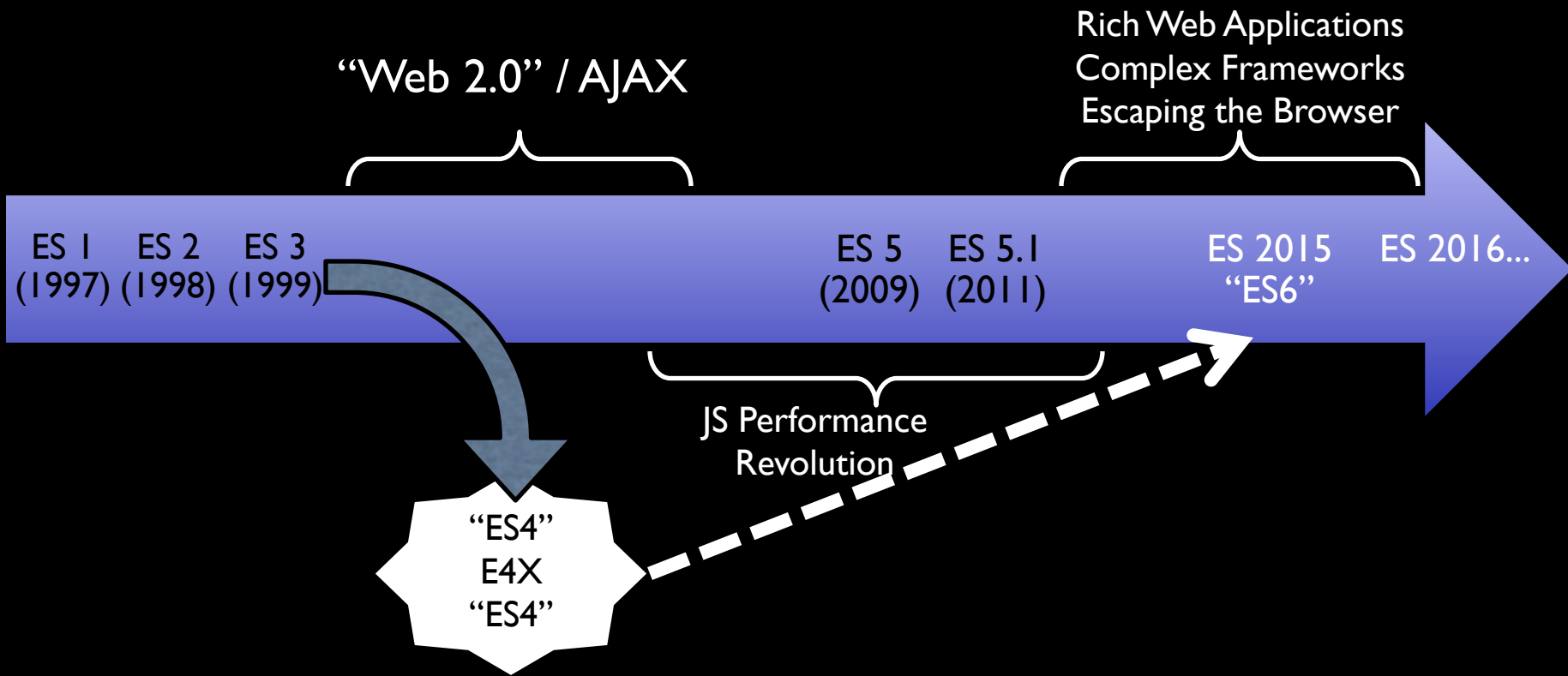
Traditional under-specification (“implementation dependent”) is bad for interoperability

Large, non-normative test suite for implementers

## 6.2.3.2 PutValue ( *V*, *W* )

1. ReturnIfAbrupt(*V*).
2. ReturnIfAbrupt(*W*).
3. If Type(*V*) is not **Reference**, throw a **ReferenceError** exception.
4. Let *base* be GetBase(*V*).
5. If IsUnresolvableReference(*V*) is **true**, then
  - a. If IsStrictReference(*V*) is **true**, then
    - i. Throw a **ReferenceError** exception.
  - b. Let *globalObj* be GetGlobalObject().
  - c. Return ? Set(*globalObj*, GetReferencedName(*V*), *W*, **false**).
6. Else if IsPropertyReference(*V*) is **true**, then
  - a. If HasPrimitiveBase(*V*) is **true**, then
    - i. Assert: In this case, *base* will never be **undefined** or **null**.
    - ii. Set *base* to ! ToObject(*base*).
  - b. Let *succeeded* be ? *base*.[[Set]](GetReferencedName(*V*), *W*, GetThisValue(*V*)).
  - c. If *succeeded* is **false** and IsStrictReference(*V*) is **true**, throw a **TypeError** exception.
  - d. Return.
7. Else *base* must be an **Environment Record**,
  - a. Return ? *base*.SetMutableBinding(GetReferencedName(*V*), *W*, IsStrictReference(*V*)) (see 8.1.1).

# The ECMAScript Standard Timeline





# TC-39 isn't like either of these



METEOR

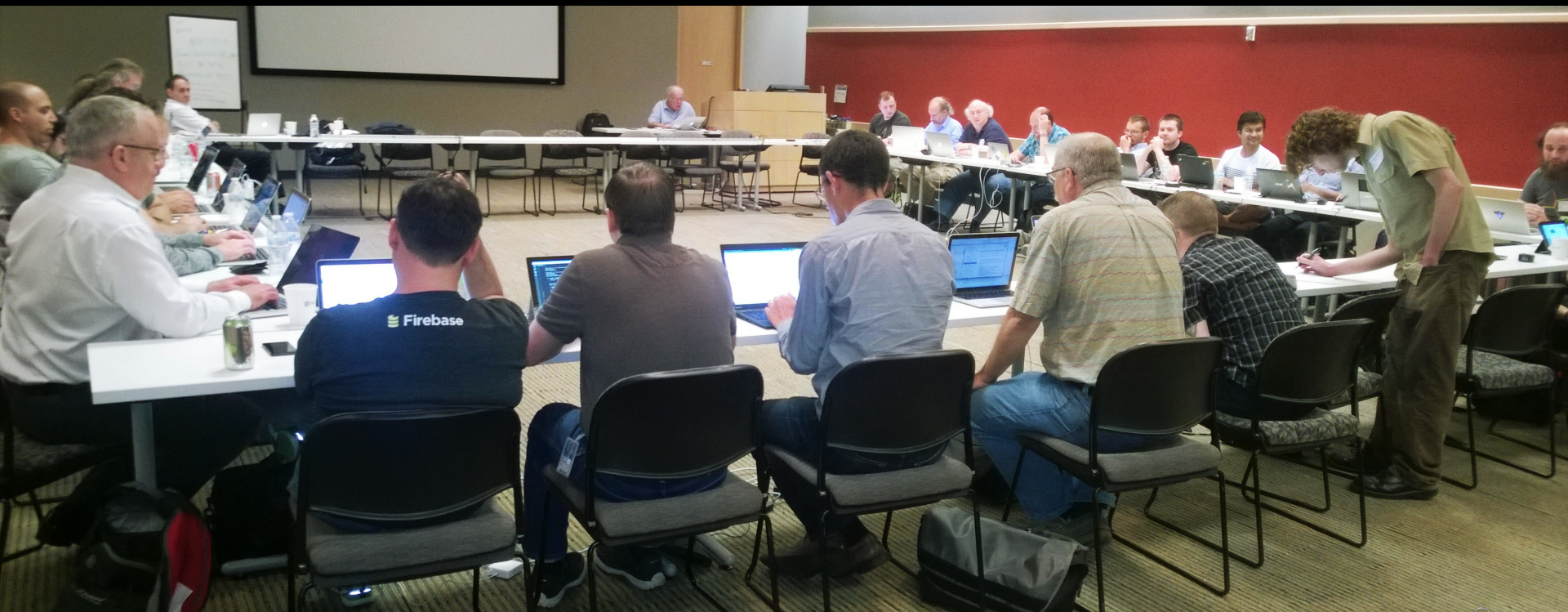


JS Foundation



NETFLIX

intel PayPal™



Google



bocoup

YAHOO!



GoDaddy





# Things TC-39 focused on for ES 2015

Modularity

Better Abstraction Capability

Better functional programming support

Better OO Support

Expressiveness and Clarity

Better Compilation Target

Things that nobody else can do

**Taking a long term perspective**

# What Kind of Language Is JavaScript?

Functional?

Object-oriented?

Class-based?

Prototype-based?

Permissive?

Secure?



Photo by crazybarefootpoet @ flickr (CC BY-NC-SA 2.0)



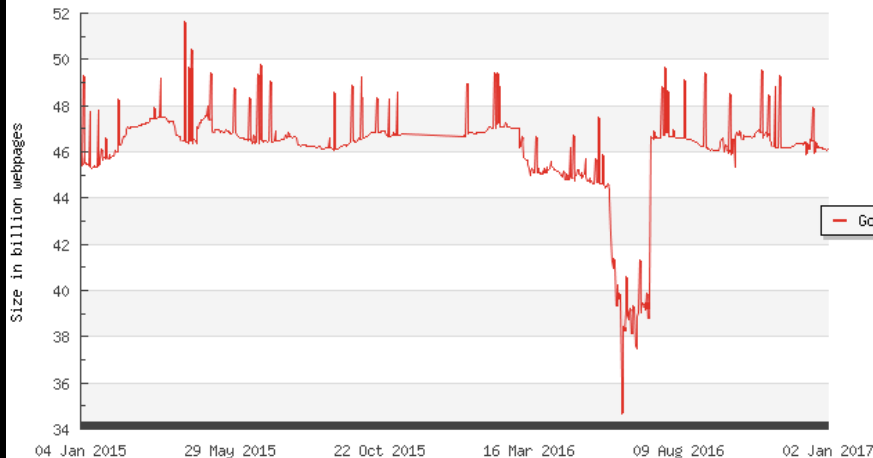


**Don't Break the Web**

# The Web is Huge

Number of web pages in Google's index

Size Google  
(Number of webpages)

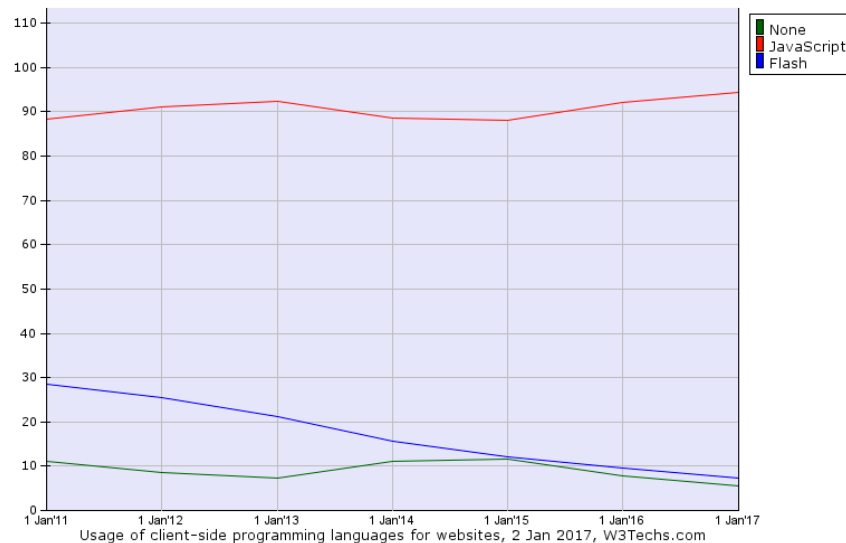


Over 45 billion web pages

<http://www.worldwidewebsize.com/>

Usage of client-side programming language for websites

The diagram shows only client-side programming languages with more than 1% usage.



Over 94% of web sites us JavaScript

[https://w3techs.com/technologies/history\\_overview/client\\_side\\_language/all](https://w3techs.com/technologies/history_overview/client_side_language/all)

# Web developers do unexpected things



Jeff Walden [:Waldo] (remove +bmo to email) 2017-01-03 11:51:57 PST

[Comment 4](#)

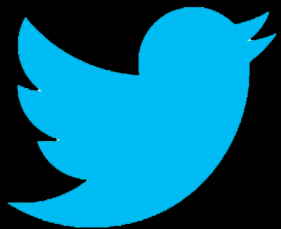
Sigh. Jira appears to be using an ancient version of the momentjs library, that improperly feature-detects Node as being anything with a "global" global property. The moment folks fixed this in <https://github.com/moment/moment/commit/1601cb1dd7b14277ba8b00cb2ece3ce637923080> which Jira seems not to have started using yet.

Unless latest Jira's updated their moment version (which seems doubtful, as you'd think they'd demo the latest version), this may end up being the straw that broke the camel's back on immediately shipping a "global" property on the global object. We can get Jira to fix, certainly, but then we'd have to wait on the rollout to customers, and I expect that would take a fairly long period of time. But who knows. I'll at least marki the dep before lunchtime. :-)





# Don't Create a Franken-language



## A common meta-tweet

ES6 **<insert some feature>** is based  
on **<insert some other language>**.



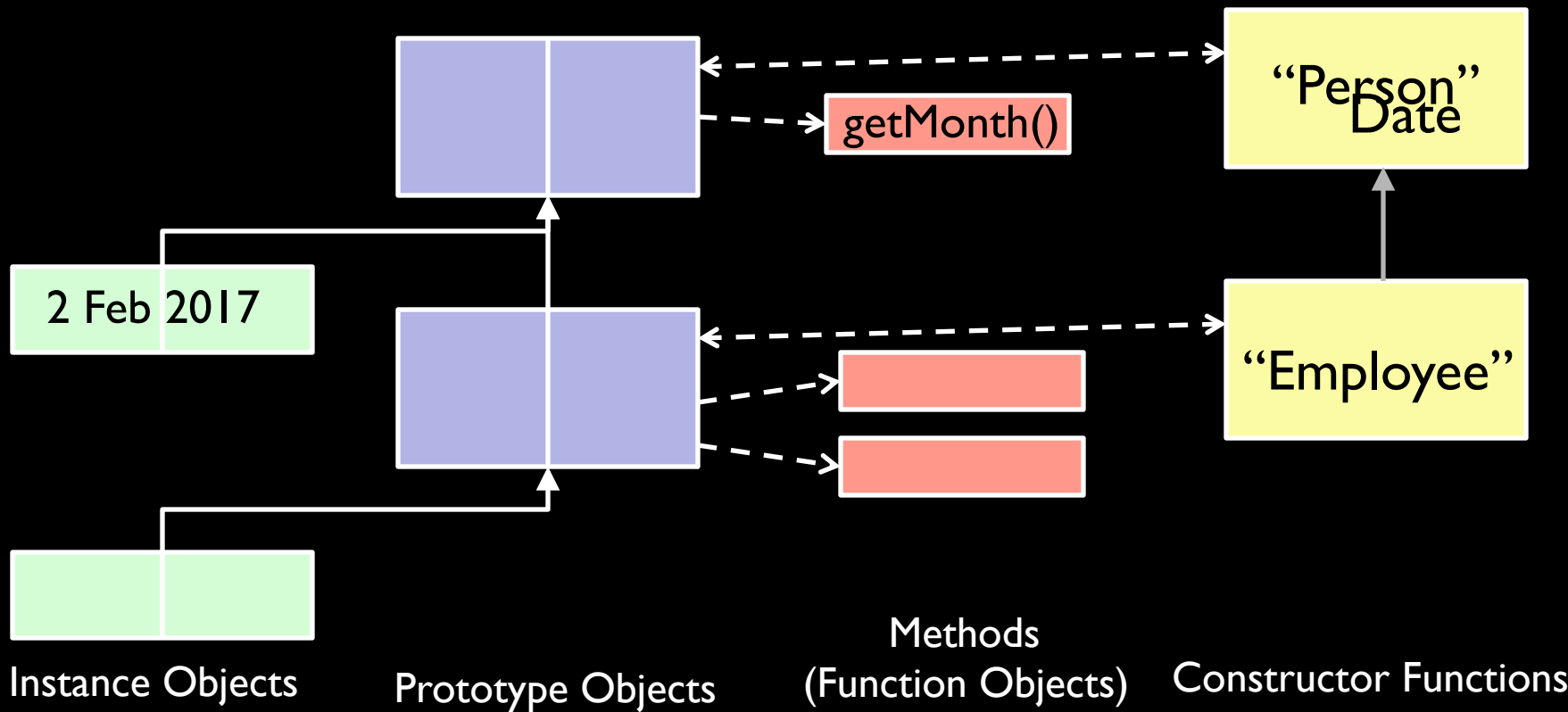
Brendan Eich 2016: Sun (represented by Bill Joy) would not have accepted [in 1995] classes, as in Java's nominal OO types, in JS. They wanted a sidekick language that did not include too much from Java itself.

<https://www.quora.com/Why-didnt-JavaScript-adopt-the-object-oriented-model-adopted-by-C++-Java-when-it-was-designed/answer/Richard-Eng-1/comment/25744373#>

# What language had the most influence on the design of ECMAScript class declarations?

- a) Java
- b) C++
- c) Ruby
- d) Dart
- e) Smalltalk
- ✓ f) Something else: JavaScript

# JavaScript Class “Constructor” Pattern



# Classes ES5 vs ES 2015

//ES5 define Employee as subclass of Person

```
function Employee(name,id) {  
  Person.call(name);  
  this.id = id;  
}  
Employee.prototype=Object.create(Person.prototype);  
Object.defineProperty(Employee.prototype, "constructor",  
  {value:Employee,enumerable:false,configurable: true});  
Employee.__proto__ = Person;  
Employee.withId = function (id) {...}  
Employee.prototype.hire = function() {...};  
Employee.prototype.fire = function () {...};  
...
```

//ES2015 define Employee as subclass of Person

```
class Employee extends Person {  
  constructor(name,id) {  
    super(name);  
    this.id = id;  
  }  
  hire () {...}  
  fire () {...}  
  static withId (id) {...}  
  ...  
}
```

Both create the same object structure



# Interconnections



# Interactions

# The closure in loop problem

```
function f(x) {  
  for (var p in x) {  
    var v = doSomething(x, p);  
    obj.addCallback(  
      function(args){  
        handle(v, p, args)}  
    );  
  }  
}  
...  
obj.runCallbacks();
```



Every callback uses the same value for v and p

# var hoisting causes the problem

```
function f(x) {  
  var p;  
  var v;  
  for (var p in x) {  
    var v = doSomething(x, p);  
    obj.setCallback(  
      function(args){  
        handle(v, p, args)}  
    );  
  }  
}  
...  
obj.runCallbacks();
```

# ES2015 could not redefine the scoping of `var`

```
function f(x) {  
  for (var p in x) {  
    var v = doSomething(x, p);  
    if (v === somethingSpecial) break;  
  }  
  if (v === somethingSpecial) ...  
}
```

# Fixing closure in loop problem: Add a new block scoped declaration

```
function f(x) {  
  for (varlet p in x) {  
    varlet v = doSomething(x, p);  
    obj.setCallback(  
      function(args){  
        handle(v, p, args)  
      })  
    };  
  }  
}  
...  
obj.runCallbacks();
```



Every callback uses a  
distinct binding for v and p



# Other local scoping WTFs

```
function f(x,x) {  
    var x;  
    for (var x in obj) {  
        if (obj[x] === somethingSpecial) {  
            var x = 0;  
            ...  
        }  
    }  
    function x() { doSomething() }  
    x();  
}
```

# Want to avoid new let WTFs

```
//duplicate declarations
```

```
function f() {
```

```
  let x = 1;
```

```
  let x = 2;
```

```
}
```

```
//duplicate let and parameter
```

```
function h(x) {
```

```
  let x = 1;
```

```
}
```

```
//duplicate let and function
```

```
function h(x) {
```

```
  let x = 1;
```

```
  function x() {}
```

```
}
```

```
//duplicate let and var
```

```
function gg() {
```

```
  let x = 1;
```

```
  var x = 2;
```

```
//hoist var to/over let
```

```
function ff() {
```

```
  let x = 1;
```

```
  if (pred) {
```

```
    var x;
```

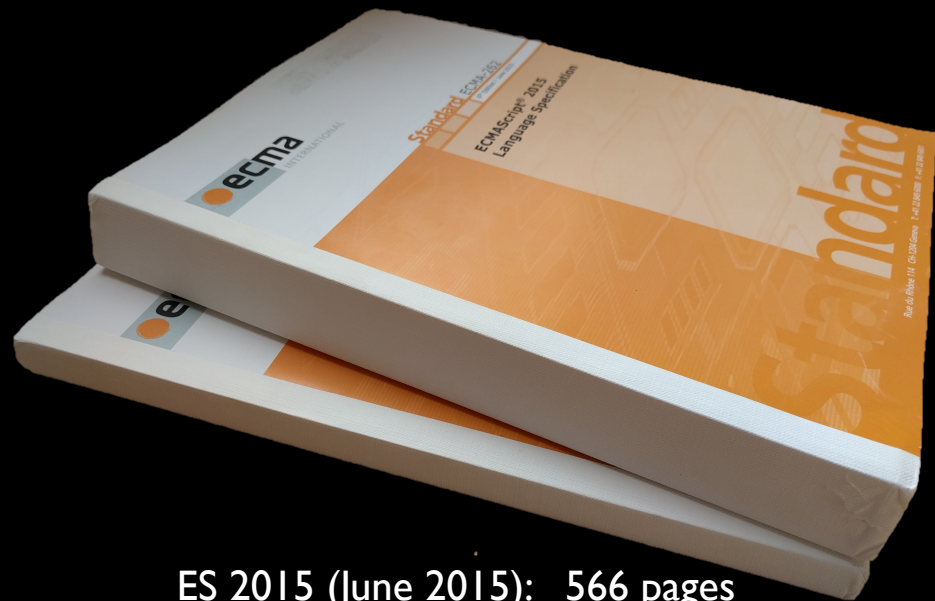
```
  }
```

```
}
```

# ECMAScript 2015:

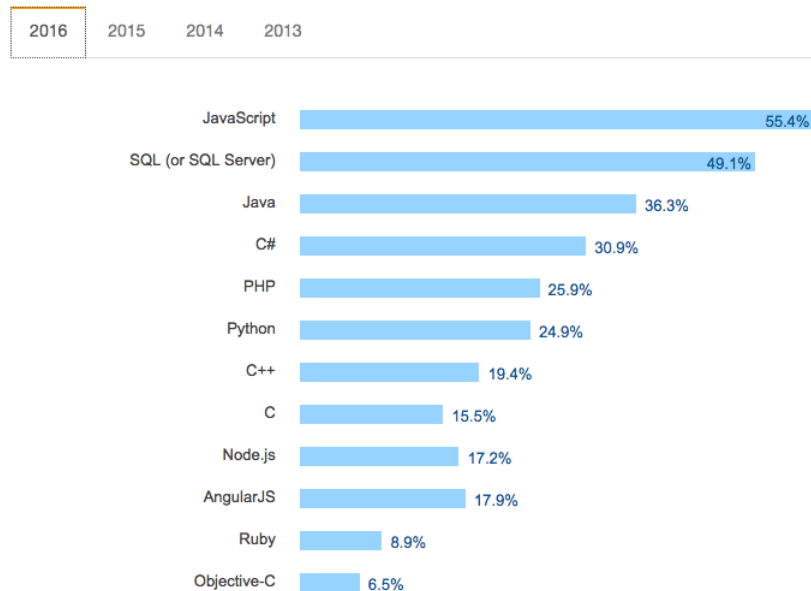
## First Comprehensive Revision Since 1999

- ✓ More concise and expressive syntax
- ✓ Modules
- ✓ Class Declarations
- ✓ Block scoped declarations
- ✓ Control abstraction via iterators and generators
- ✓ Promises
- ✓ String interpolation/Internal DSL support
- ✓ Subclassable built-ins
- ✓ Binary Array Objects with Array methods
- ✓ Built-in hash Maps and Sets + weak variants
- ✓ More built-in Math and String functions
- ✓ Improved Unicode support, full Unicode RegExp
- ✓ Async function (2017)



ES 2015 (June 2015):	566 pages
ES 5 (Dec. 2009):	252 pages
ES 3 (Dec. 1999):	188 pages
ES 2 (Aug 1998):	117 pages
ES 1 (June 1997):	110 pages

## I. Most Popular Technologies



49,397 responses

More people use JavaScript than use any other programming language. PHP appears to be falling out of favor as Node and Angular emerge.

Stack Overflow 2016 Developer Survey

<http://stackoverflow.com/research/developer-survey-2016>

# 2016's most popular programming language: JavaScript

1 JavaScript

2 Java

3 PHP

4 Python

5 C#

5 C++

5 Ruby

8 CSS

9 C

10 Objective-C

RedMonk Top 10  
Programming Languages  
January 2016

<http://redmonk.com/sograzy/2016/02/19/language-rankings-1-16/>

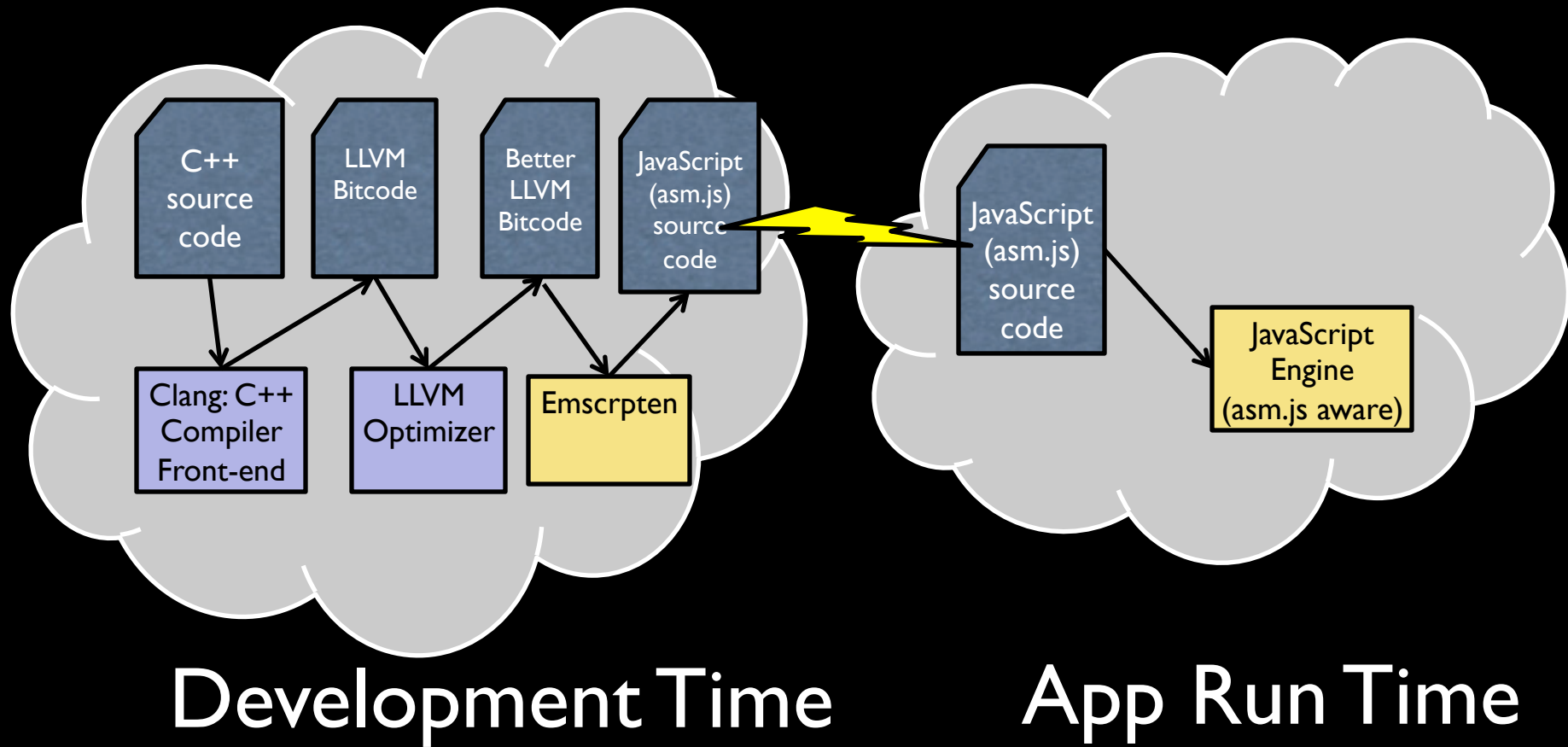
# JavaScript is the Browser VM

## “Transpilers”

Wikipedia: a type of compiler that takes the source code of a program written in one programming language as its input and produces the equivalent source code in another programming language.

Babel, TypeScript, Dart, Flow, CoffeeScript, ...

# C++ to JavaScript



# JavaScript on the Server



An asynchronous event driven JavaScript runtime, node.js<sup>†</sup> is designed to build scalable network applications.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

“over 3.5 million users and an annual growth rate of 100 percent”

<http://nodejs.org/>

“Average downloads per day (2015): 266,472”

<https://nodesource.com/assets/blog/node-by-numbers/node-by-numbers.pdf>

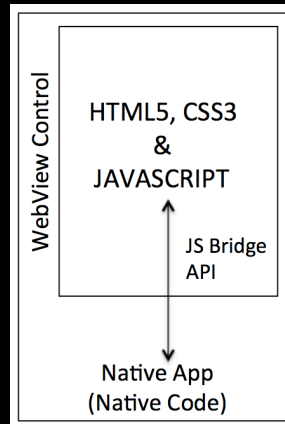




# Non-web Interactive Apps

WebViews/Hybrid Mobile Apps

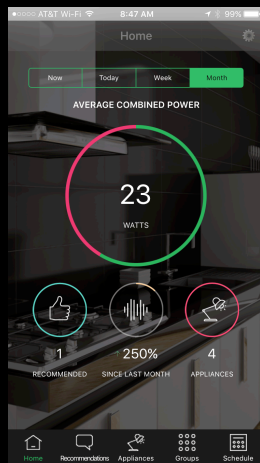
NativeScript<sup>†</sup>, ReactNative

Electron<sup>†</sup>

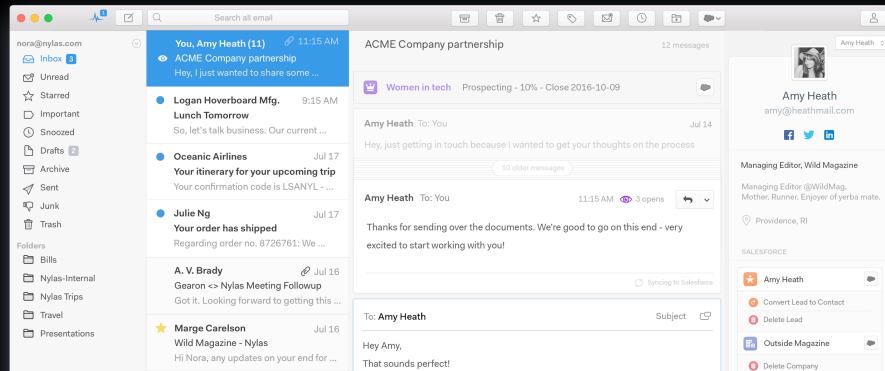


Mobile Platform/OS	WebView Component Name
Apple 	<b>UIWebView</b> (iOS 4+) <b>WKWebView</b> (iOS 8.1+)
Android 	<b>WebView</b> (With Chromium's Rendering Engine used from Android Kitkat 4.4+) <b>Updatable WebView</b> (Android Lollipop 5+)

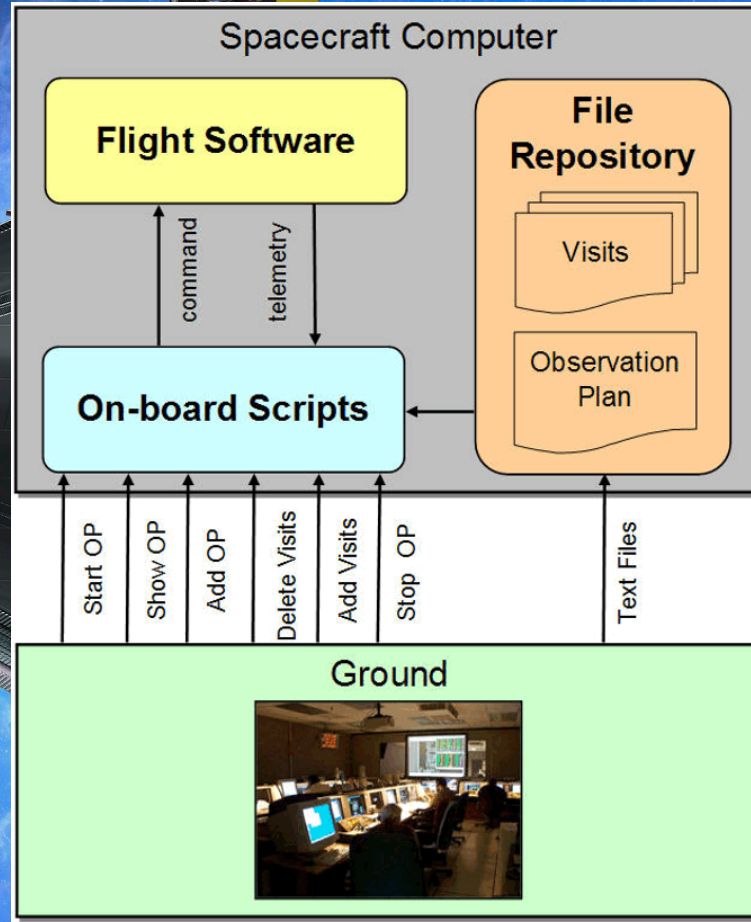
<http://appsonmob.com/hybrid-app-webview-performance-ios-android/>



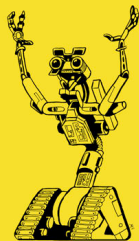
<https://www.nativescript.org/showcases#mewatt>



<https://www.nylas.com/>



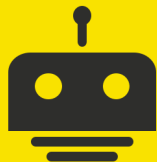
# JavaScript for Devices/ Embedded/Robotics



Johnny-Five<sup>†</sup>

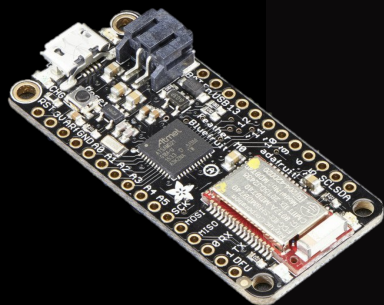
<http://johnny-five.io/>

<http://nodebots.io/>



NodeBots

Robots powered by JavaScript



<https://burningservos.com/2016/12/19/walking/>





"Javascript is the new C"

realtime applications, C's real value emerged from that ubiquity: C was the only truly platform-independent programming language. By 1990, you could write a C program and run it on any computer in existence.

Today Javascript has taken over that mantel. C/C++ is no longer universal. *Only Javascript will run on Windows, Linux, OS X, iOS and Android platforms.*

[REST API on a Pi, I have control over your GPIO I/O ports over the internet](#)

[How To Install Windows 10 IoT on a Raspberry Pi](#)

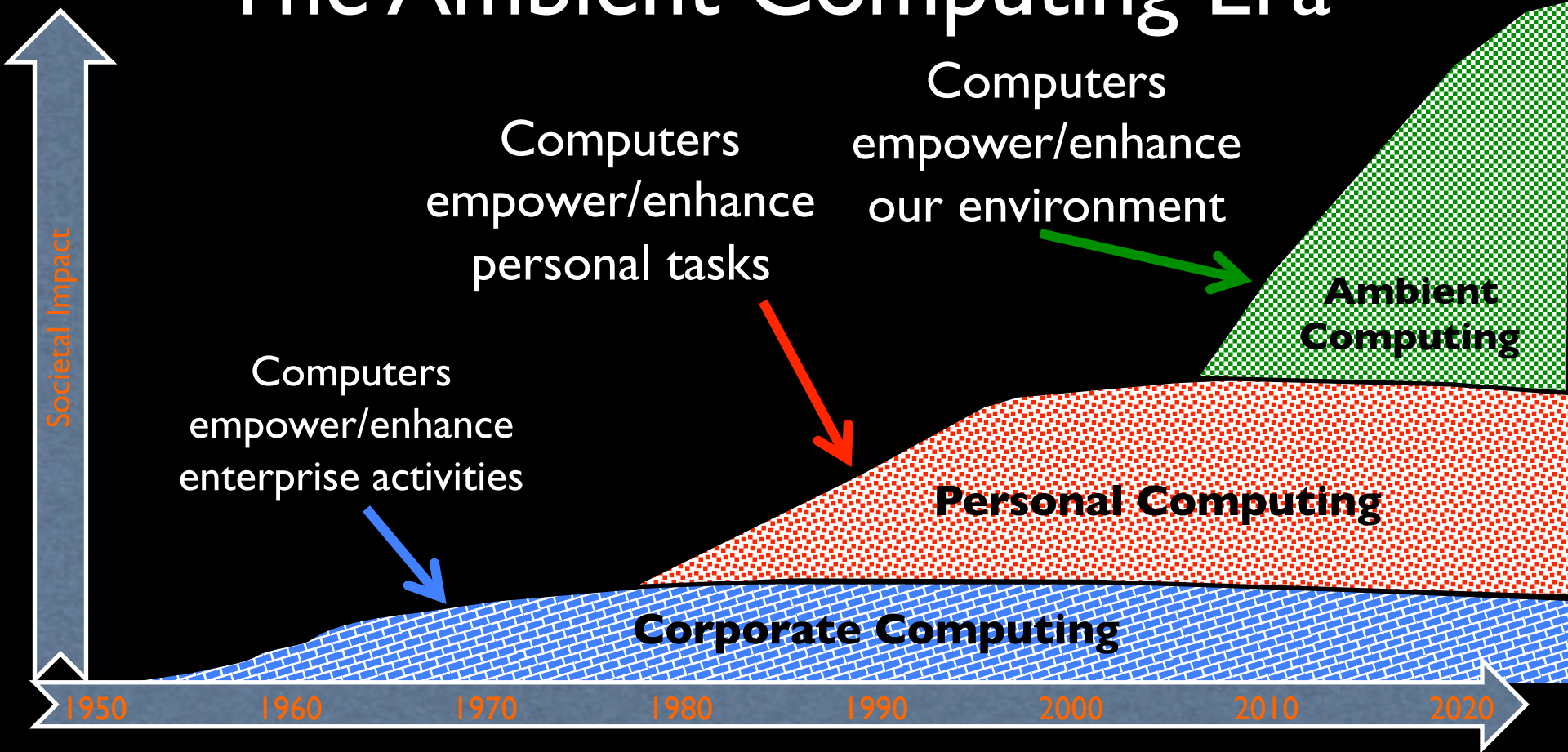
as an extravagance, C was a revelation. Near-assembler fast, bit-level operations, but still an expressive 3GL.

OK, so nothing like JavaScript then?

Wait, that's only the beginning of the story.

```
CMF  AX,122    ; compare AX to 122
JG   DONE      ; if greater, jump
SUB  AX,32      ; subtract 32 from AX
DONE: RET       ; return to caller
SUB32 ENDP      ; procedure ends
```

# The Ambient Computing Era



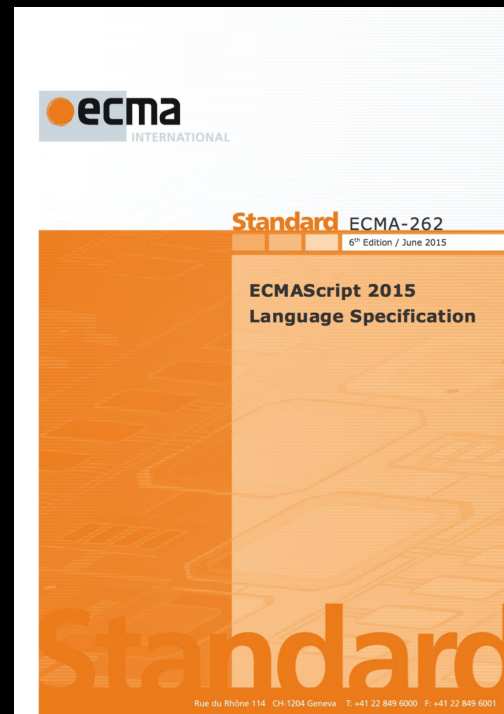
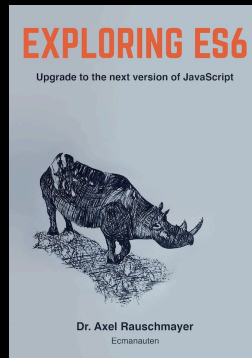


# Each Computing Era has had Canonical Programming Languages

Corporate Computing Era – COBOL/Fortran

Personal Computing Era – C/C++ family

JavaScript: The Canonical Language of the Ambient Computing Era?



Allen Wirfs-Brock

<http://www.wirfs-brock.com/allen>

[allen@wirfs-brock.com](mailto:allen@wirfs-brock.com)

@awbjs