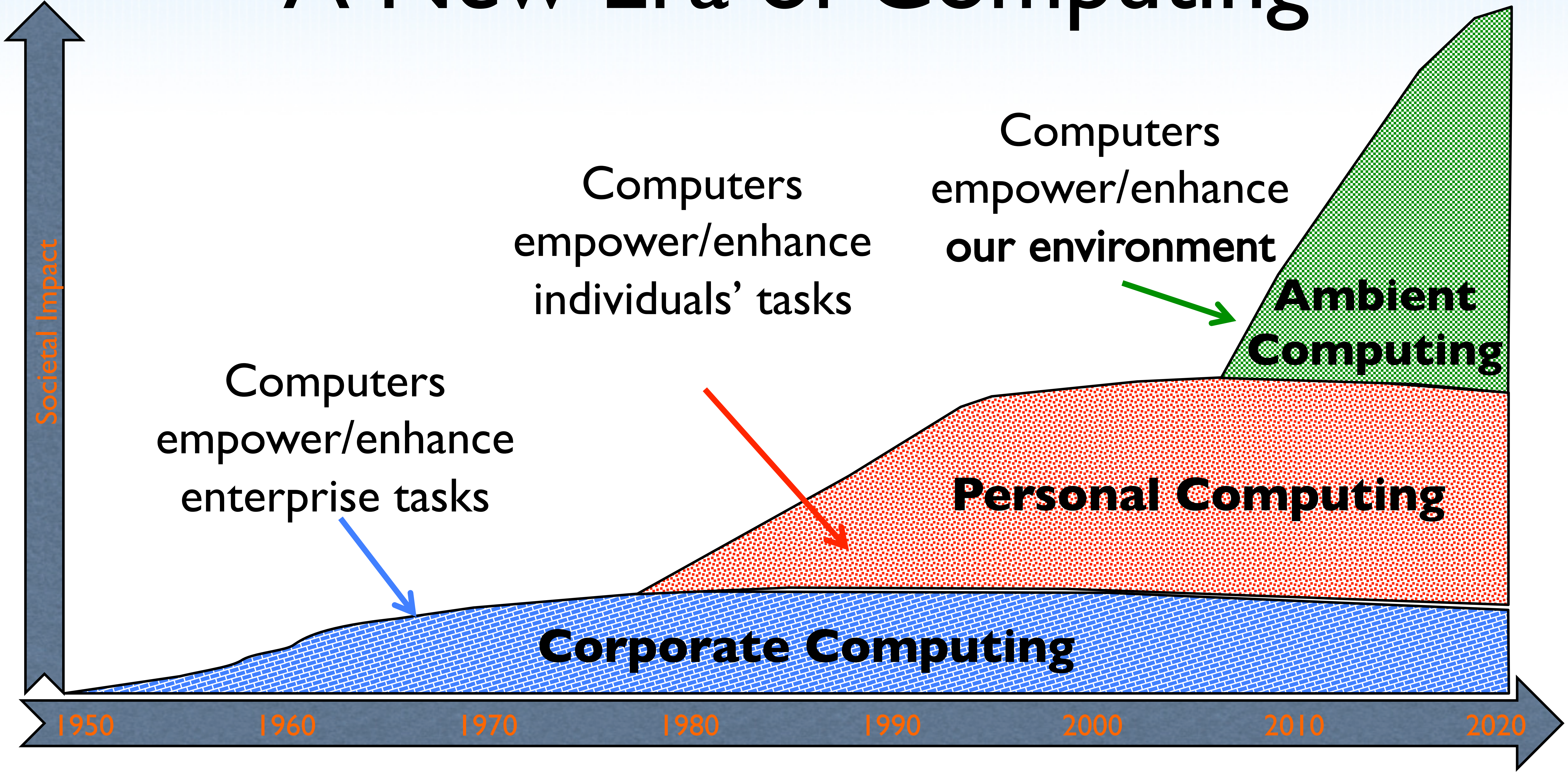
JavaScript Today and Tomorrow:
Evolving the Ambient Language of the Ambient Computing Era

Talk, by Allen Wirfs-Brock
Mozilla
@awbjs

QCON London 2011, March, 2012

As we leave the personal computing era and rapidly enter the era of ambient computing, JavaScript's position as the dominant programming language is becoming increasingly apparent. JavaScript isn't just a language for directly writing web applications, it is also rapidly becoming the virtual machine and compilation target for every other languages that needs to supports the ambient web application platform. In this talk I'll take a look at the current status of JavaScript from both these perspectives and examine some of its strengths and weakness. I'll explain how JavaScript implementers work together to ensure interoperable implementations. I'll also explain how the JavaScript standardization process works to introduce new features into the JavaScript language and what changes we can reasonably expect to see in the future.

A New Era of Computing



The Ambient Computing Era

- Devices not Computers
- Ubiquitous access to information
- Computing augmented life

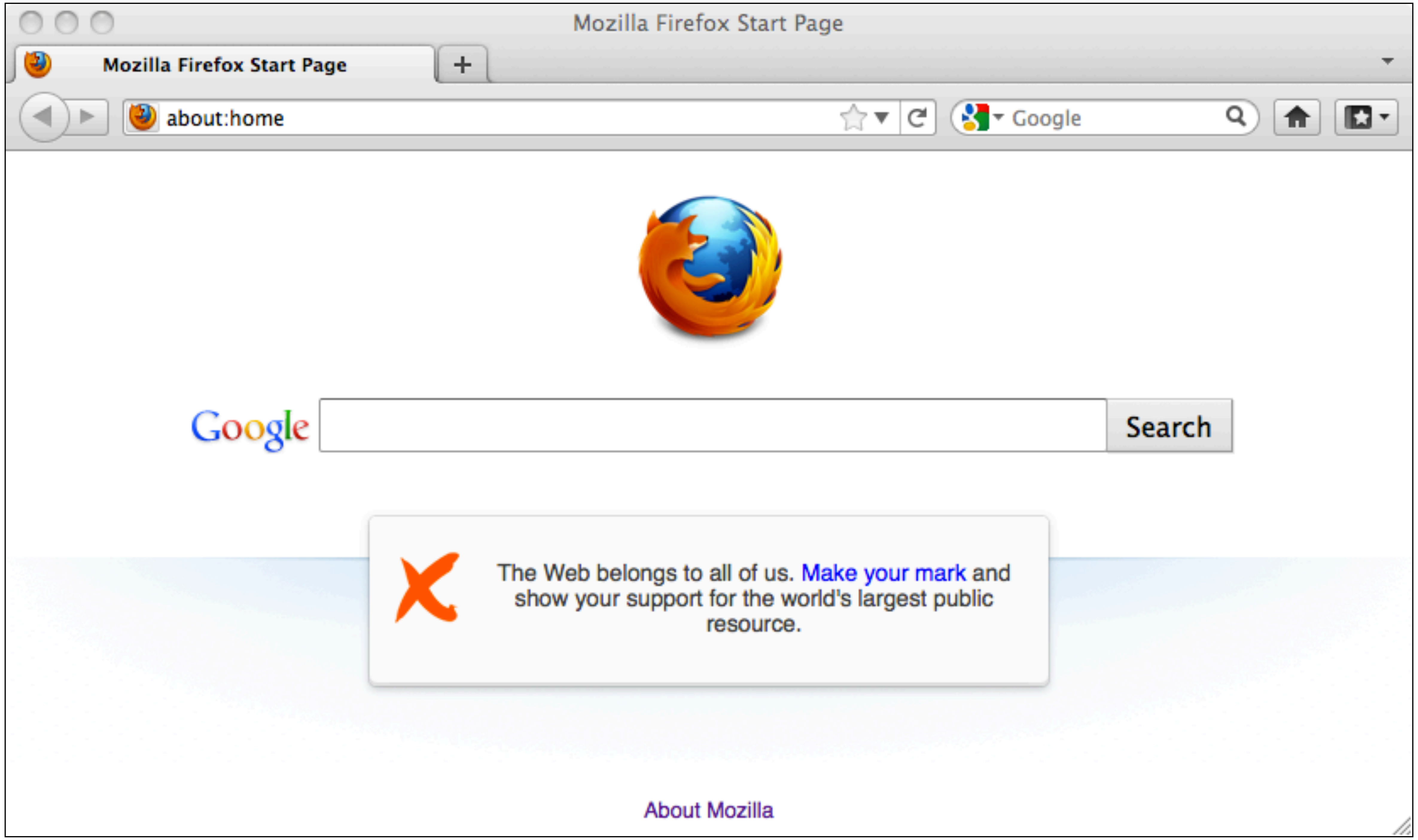
Computers enhance the world I live in. I need my stuff (data and apps) right now, wherever I am, using whatever device is available. I can't live without it!

Every Computing Era Has a Dominant Application Platfrom

- Corporate Computing Era: IBM Mainframes
- Personal Conmputing Era: Microsoft/Intel PC
- Ambient Computing Era: T.B.D (or is it?)

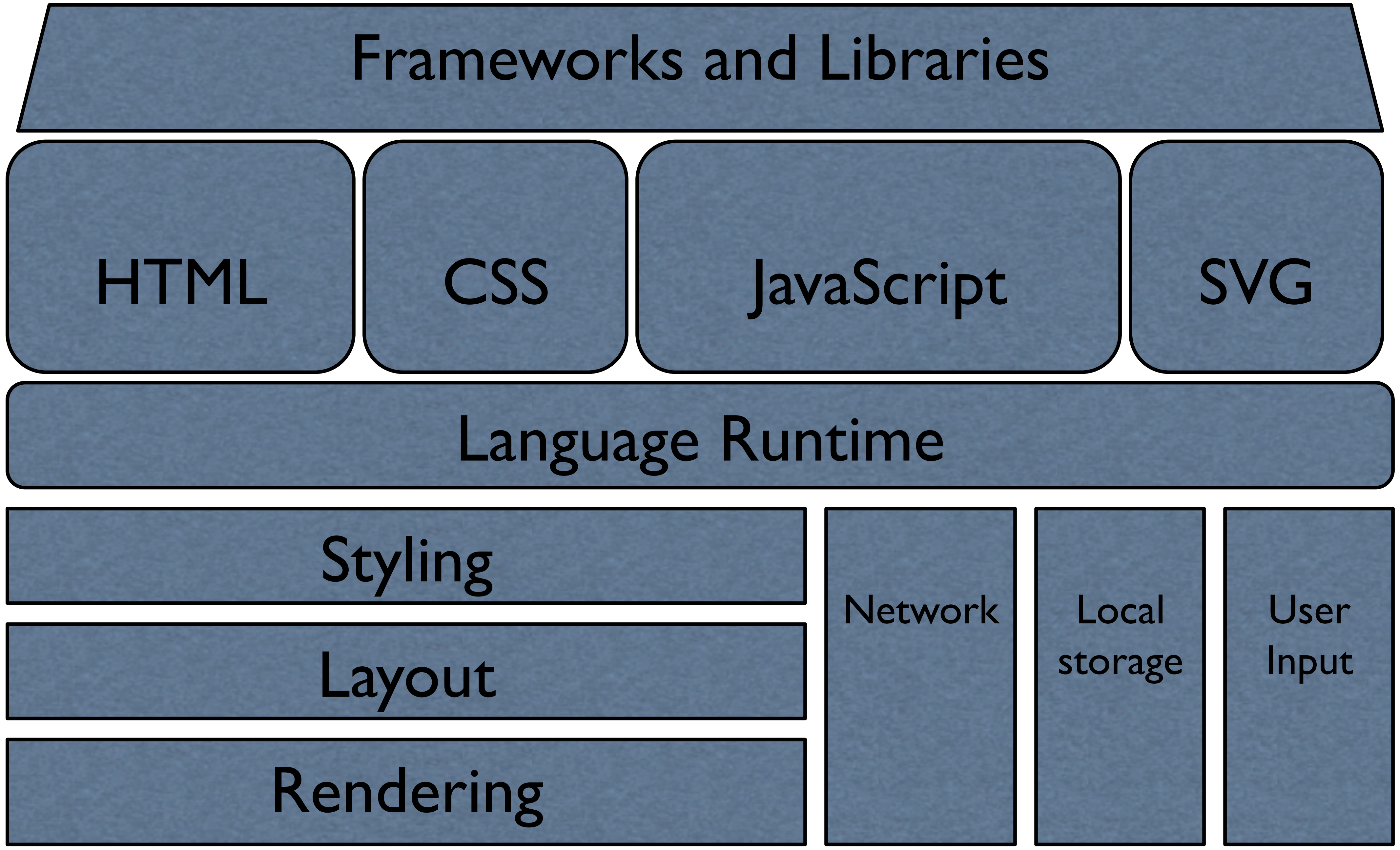
Created by Market Demand,
"Good Enough" Technical Foundation,
and Superior Business Execution

What do you have when you strip away the PC application part of a web browser?



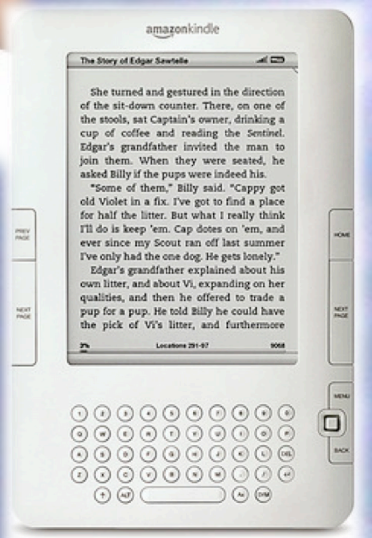
The Web belongs to all of us. Make your mark and show your support for the world's largest public resource.

What do you have when you strip away the PC application part of a web browser?

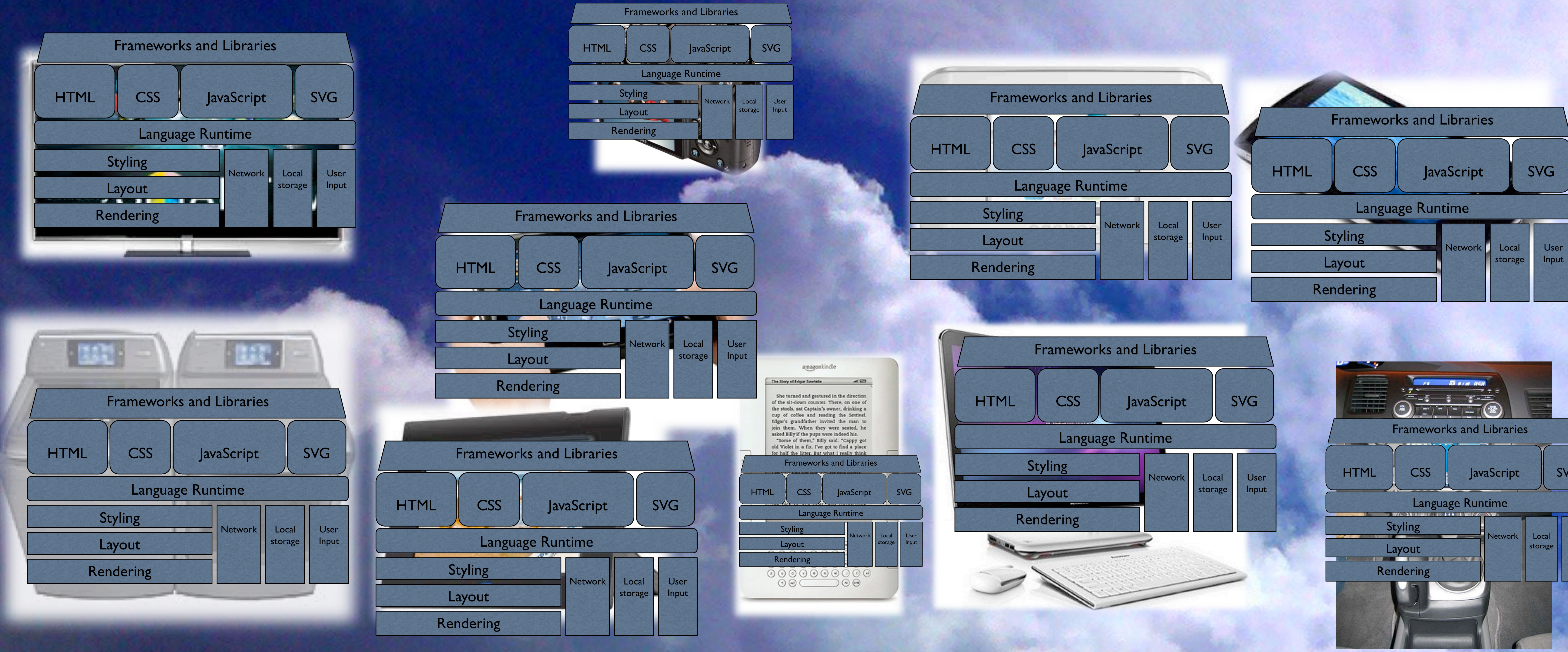


The Web Application Platform

The Web is the Platform

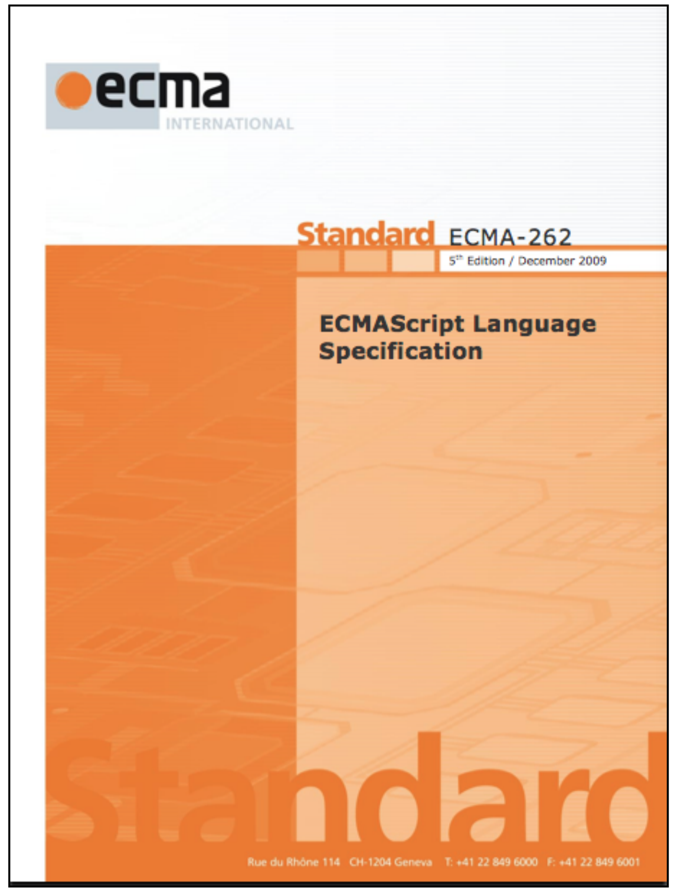


The Web is the Platform



Each Computing Era has had Canonical Programming Languages

- Corporate Computing Era – COBOL/Fortran
- Personal Computing Era – C/C++ family
- Ambient Computing Era – JavaScript ??


ecma
INTERNATIONAL
Standard ECMA-262
5th Edition / December 2009
ECMAScript Language Specification
Standard

Why JavaScript?

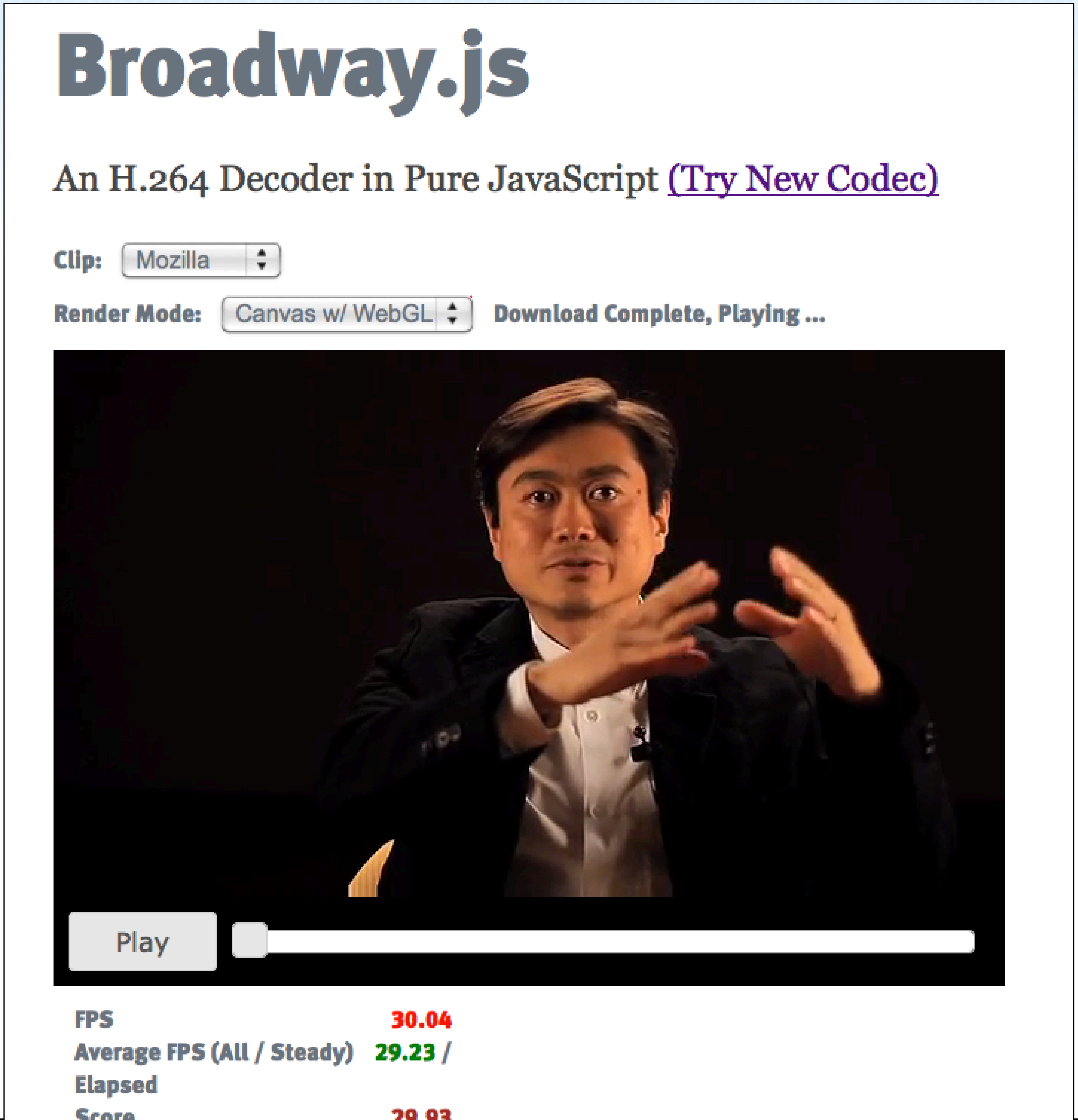
Because "Worse is Better"

Dick Grabriel
http://www.dreamsongs.com/WorseIsBetter.html

- It's there – It's working
- It's good enough
- It's getting better
- What could replace it?
- How could that happen?

http://odetocode.com/Blogs/scott/archive/2009/03/18/signs-that-your-javascript-skills-need-updating.aspx

JavaScript Performance:



Broadway.js

An H.264 Decoder in Pure JavaScript (Try New Codec)

Clip: Mozilla

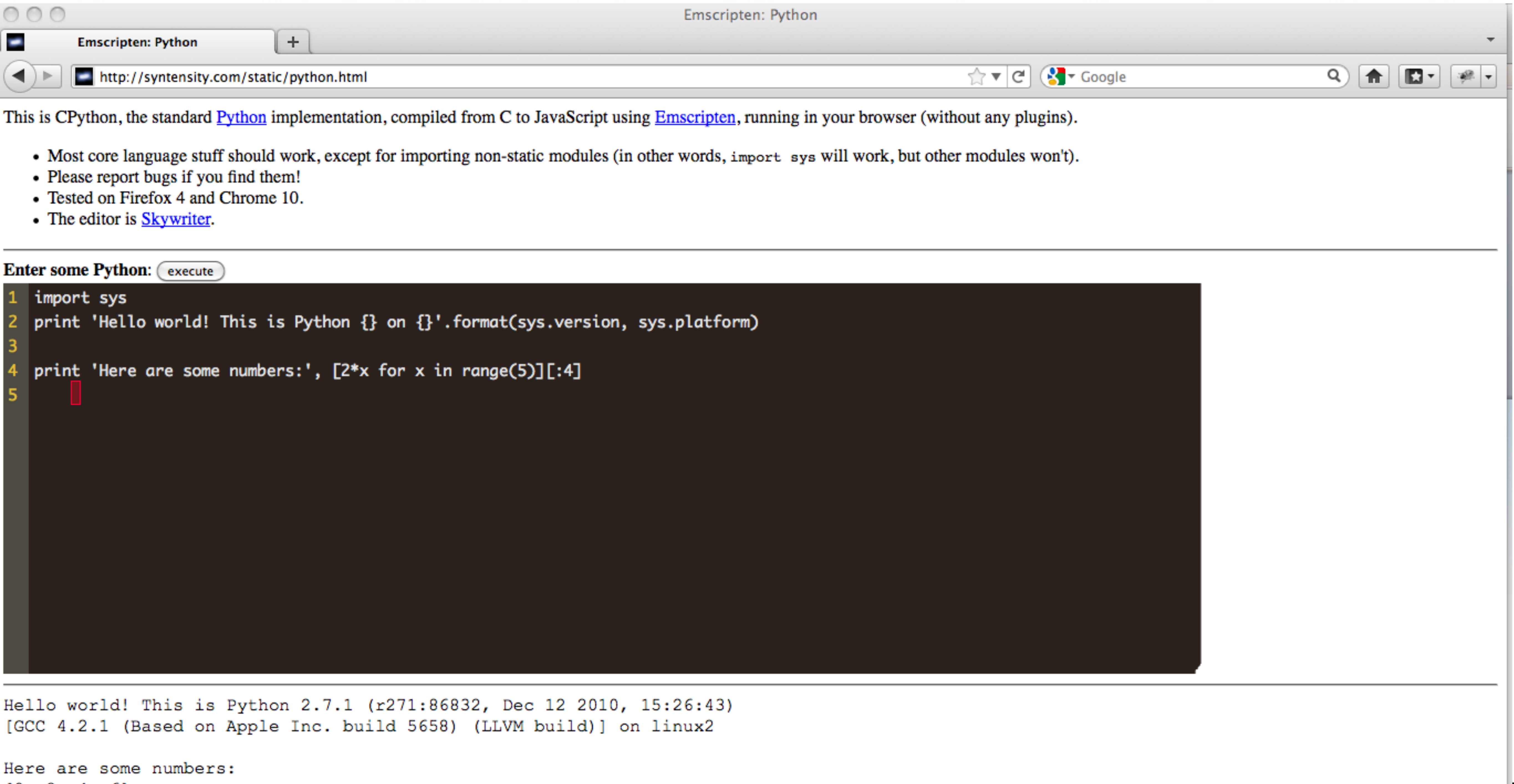
Render Mode: Canvas w/ WebGL Download Complete, Playing ...

```
FPS                          30.04
Average FPS (All / Steady)   29.23 /
Elapsed
Score                        29.93
```

http://mbebenita.github.com/Broadway/broadway.html

http://haxpath.squarespace.com/imported-20100930232226/2011/10/28/broadwayjs-h264-in-javascript.html

Emscripten
Compiling C/C++ to JavaScript



This is CPython, the standard Python implementation, compiled from C to JavaScript using Emscripten, running in your browser (without any plugins).

- Most core language stuff should work, except for importing non-static modules (in other words, `import sys` will work, but other modules won't).
- Please report bugs if you find them!
- Tested on Firefox 4 and Chrome 10.
- The editor is Skywriter.

Enter some Python:

```
import sys
print 'Hello world! This is Python {} on {}'.format(sys.version, sys.platform)

print 'Here are some numbers:', [2*x for x in range(5)][:4]
```

```
Hello world! This is Python 2.7.1 (r271:86832, Dec 12 2010, 15:26:43)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build)] on linux2

Here are some numbers:
```

Tranlates LLVM intermediate code to JavaScript

https://github.com/kripken/emscripten/wiki

http://syntensity.com/static/python.html

Langauges That Compile To JavaScript

- CoffeeScript (“Improved” JavaScript syntax)
- Java (GWT)
- Script# (C# dialect)
- Dart
- ClojureScript (Lisp dialect)
- Dozens more listed at
  https://github.com/jashkenas/coffee-script/wiki/List-of-languages-that-compile-to-JS

JavaScript is the “virtual machine” of the Web Platform.

How is JavaScript Used Today?

- Still lots of Web 1.0/2.0 style DOM manipulation
  - Small program fragments operating upon pre-existing domain models
- It is being increasing used as a the primary programming language for rich applications on both client and server platforms.
  - Large and often complex programs that define their own domain models

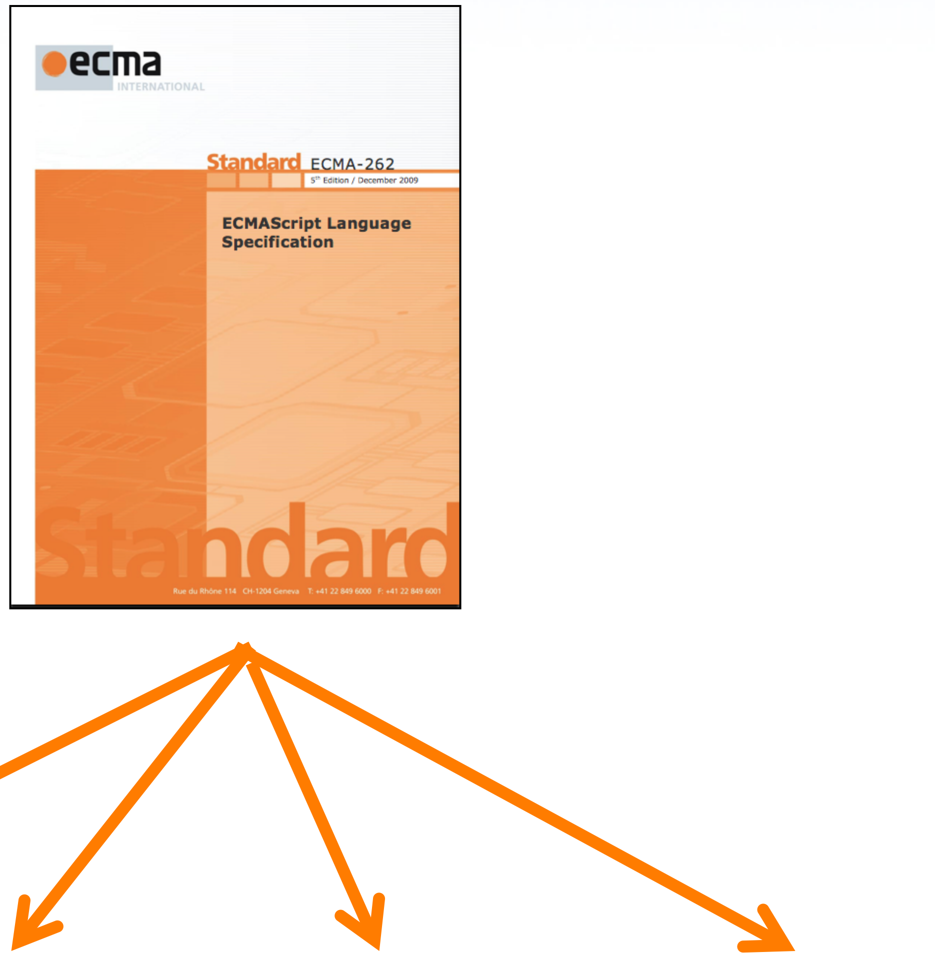
JavaScript must evolve to better support this new usage pattern.

How Does JavaScript Evolve

- No single vendor controls “JavaScript”
- What drives innovation?
- What actually gets implemented?
- Who makes language design decisions?
- Cumbersome standardization processes
- Too slow rate of change? … or too fast?

What is ECMAScript?

- ECMAScript is the name of the international standard that defines JavaScript
- Developed by Technical Committee 39 (TC-39) of Ecma International
- Issued as a Ecma-262 and ISO/IEC 16262
- Not part of W3C


ecma INTERNATIONAL

Standard ECMA-262

5th Edition / December 2009

ECMAScript Language Specification

Standard

Google V8 | Mozilla SpiderMonkey | Microsoft Chakra | Webkit JSCore

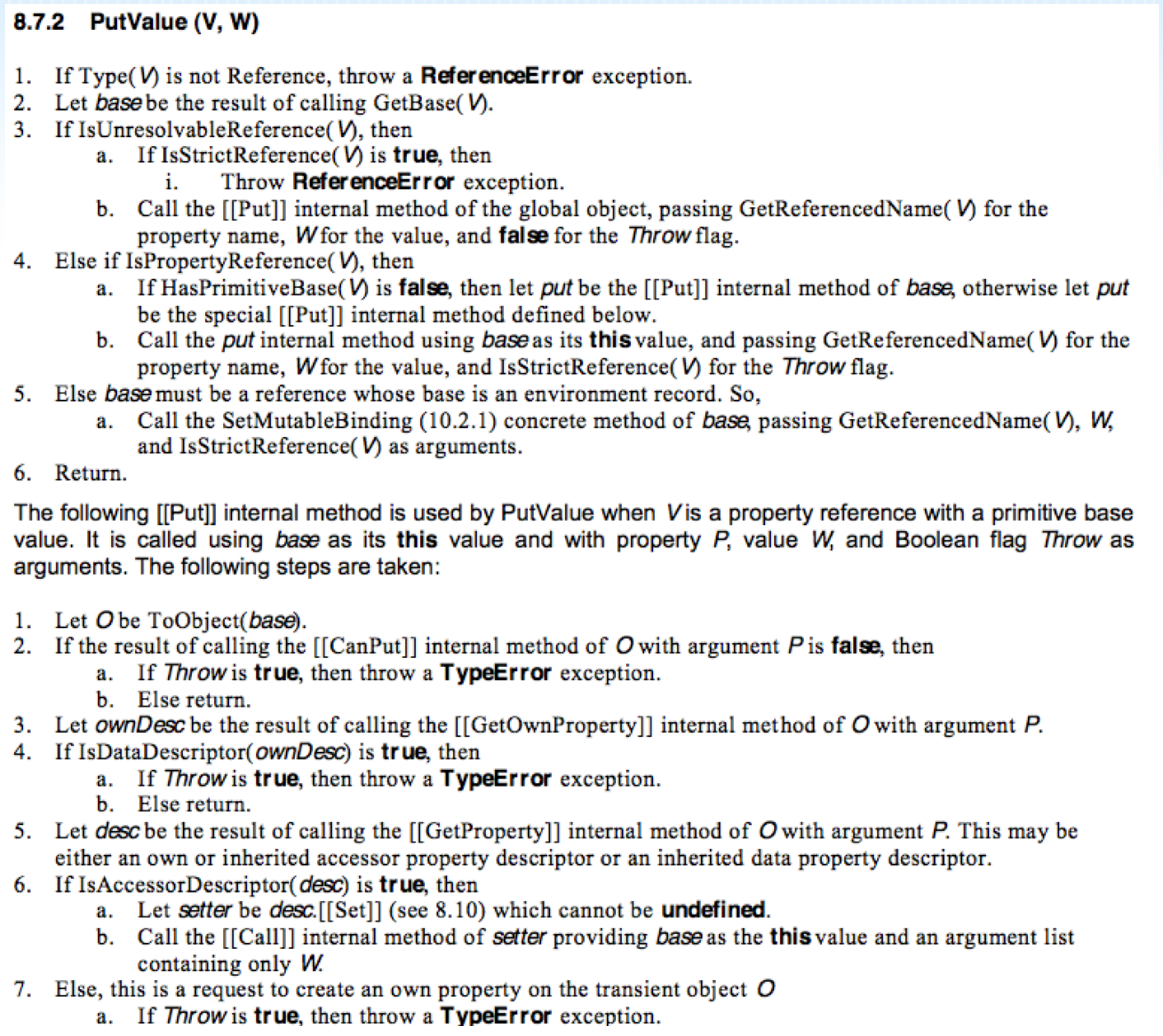
JavaScript Implementations

Interoperability is TC-39's highest priority

- A detailed and highly prescriptive algorithmic specification
- Large, non-normative test suite for implementers

ecmascript test262

http://test262.ecmascript.org/



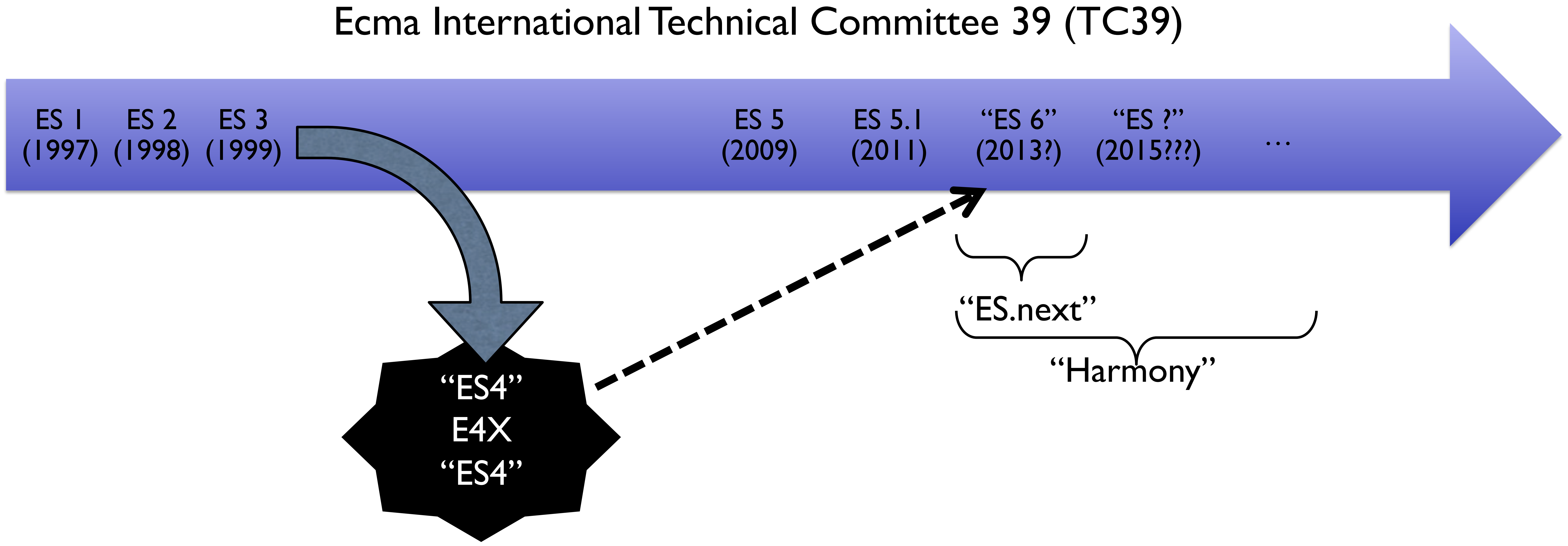
8.7.2 PutValue (V, W)

```
1. If Type(V) is not Reference, throw a ReferenceError exception.
2. Let base be the result of calling GetBase(V).
3. If IsUnresolvableReference(V), then
    a. If IsStrictReference(V) is true, then
        i. Throw ReferenceError exception.
    b. Call the [[Put]] internal method of the global object, passing GetReferencedName(V) for the
       property name, W for the value, and false for the Throw flag.
4. Else if IsPropertyReference(V), then
    a. If HasPrimitiveBase(V) is false, then let put be the [[Put]] internal method of base, otherwise let put
       be the special [[Put]] internal method defined below.
    b. Call the put internal method using base as its this value, and passing GetReferencedName(V) for the
       property name, W for the value, and IsStrictReference(V) for the Throw flag.
5. Else base must be a reference whose base is an environment record. So,
    a. Call the SetMutableBinding (10.2.1) concrete method of base, passing GetReferencedName(V), W,
       and IsStrictReference(V) as arguments.
6. Return.
```

The following [[Put]] internal method is used by PutValue when *V* is a property reference with a primitive base value. It is called using *base* as its **this** value and with property *P*, value *W*, and Boolean flag *Throw* as arguments. The following steps are taken:

```
1. Let O be ToObject(base).
2. If the result of calling the [[CanPut]] internal method of O with argument P is false, then
    a. If Throw is true, then throw a TypeError exception.
    b. Else return.
3. Let ownDesc be the result of calling the [[GetOwnProperty]] internal method of O with argument P.
4. If IsDataDescriptor(ownDesc) is true, then
    a. If Throw is true, then throw a TypeError exception.
    b. Else return.
5. Let desc be the result of calling the [[GetProperty]] internal method of O with argument P. This may be
   either an own or inherited accessor property descriptor or an inherited data property descriptor.
6. If IsAccessorDescriptor(desc) is true, then
    a. Let setter be desc.[[Set]] (see 8.10) which cannot be undefined.
    b. Call the [[Call]] internal method of setter providing base as the this value and an argument list
       containing only W.
7. Else, this is a request to create an own property on the transient object O
    a. If Throw is true, then throw a TypeError exception.
```

The ECMAScript Standard Timeline

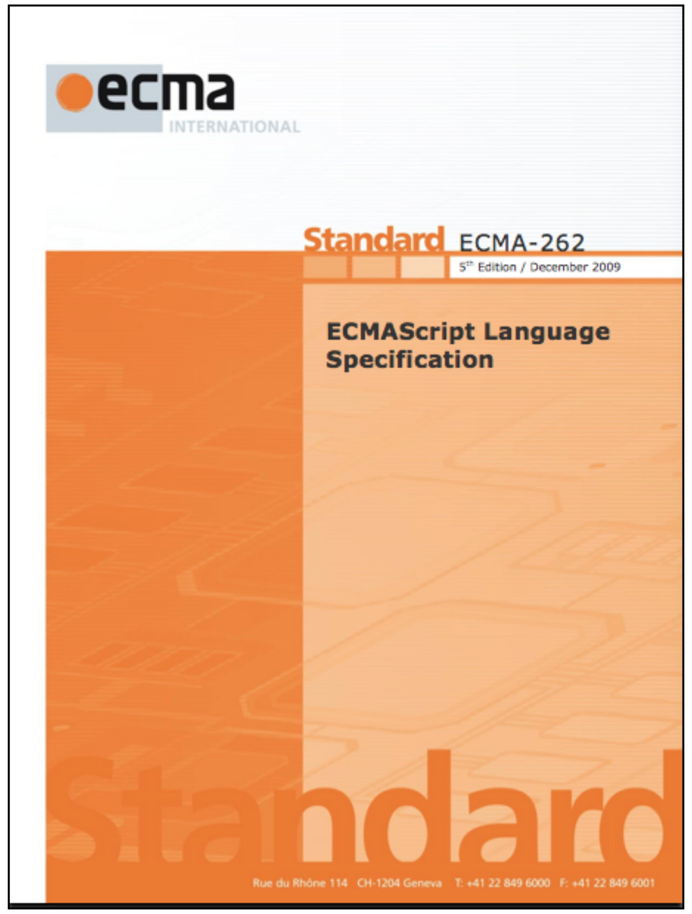


What did ES 5 Accomplish

- Re-established a viable standards process for JavaScript
- Brought IE (JScript) into conformance
- Standardized a small number of important enhancements

What Did ES5 Add to JavaScript

- JSON generation and parsing functions
- Accessor (getter/setter) methods
- ISO date processing
- “Array Extras” methods
- Object.create, Function bind
- Property Attribute control
- Object “lock-down”
- Strict mode



ecma INTERNATIONAL

Standard ECMA-262

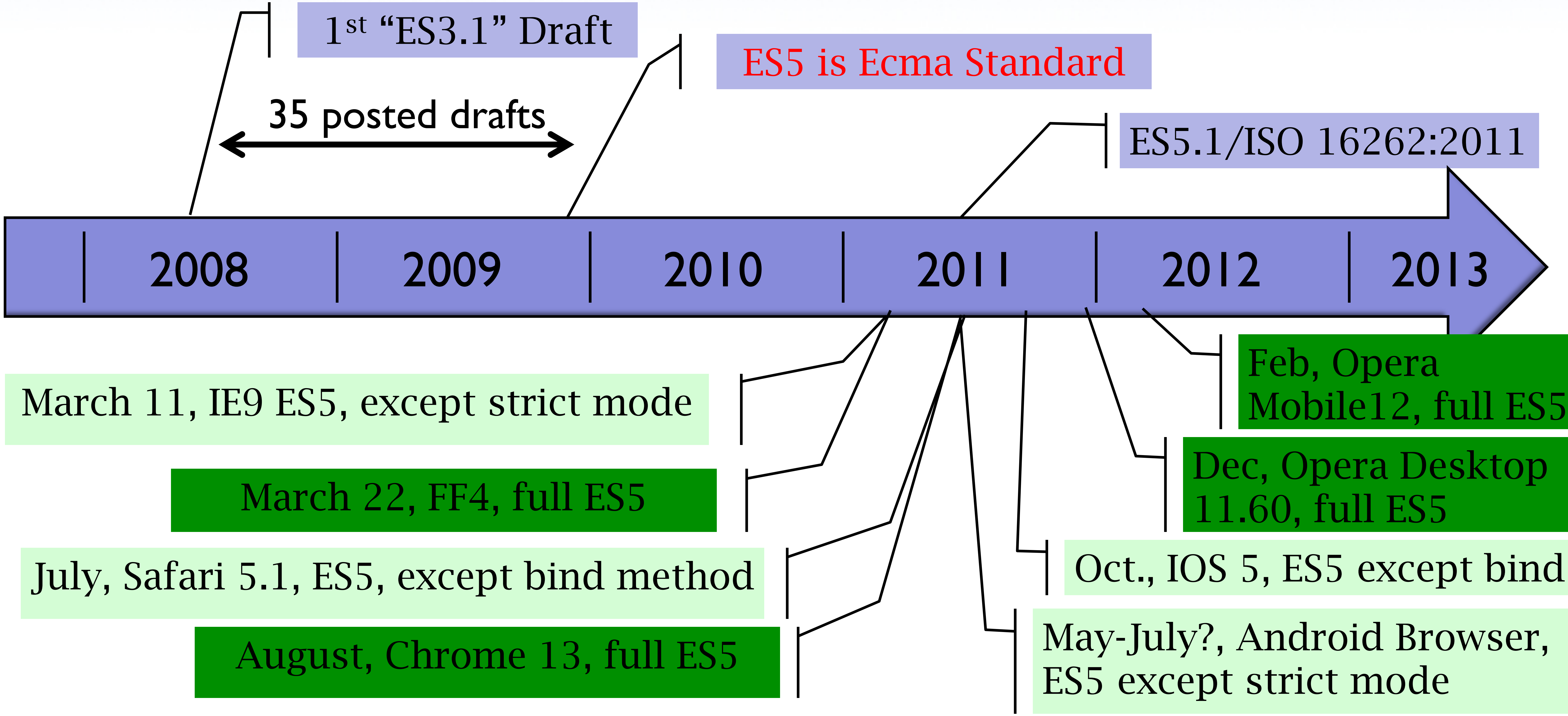
5th Edition / December 2009

ECMAScript Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

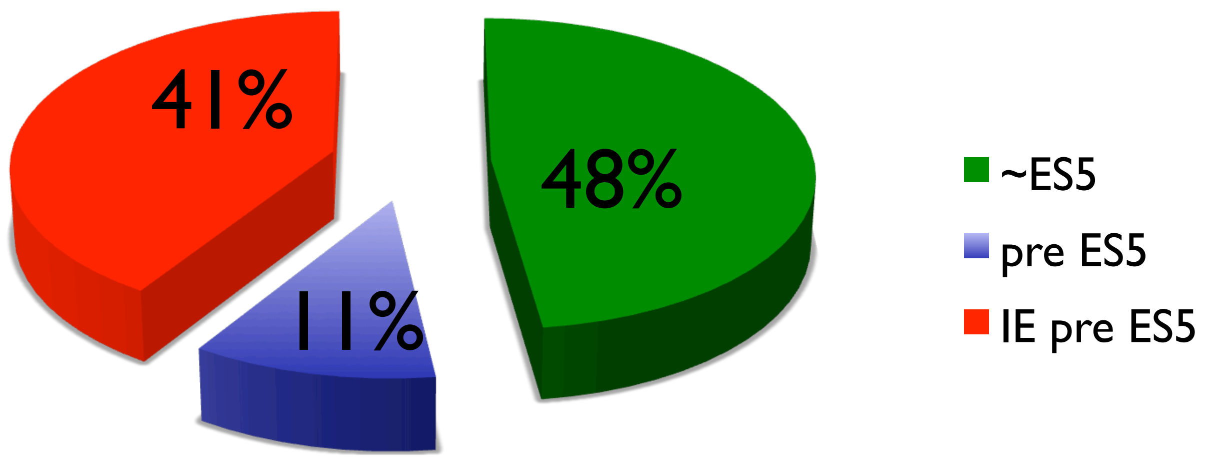
Revising a Standard Takes Time



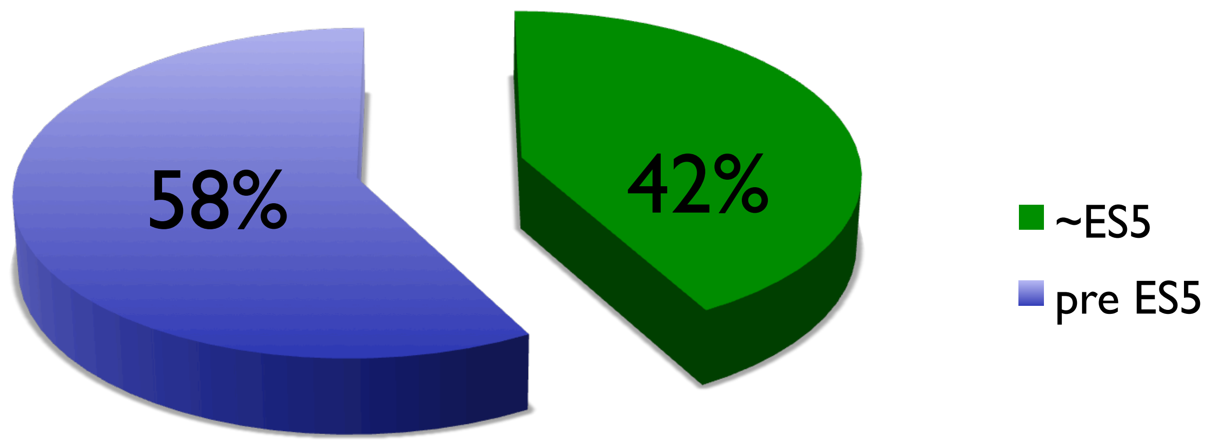
1st "ES3.1" Draft
ES5 is Ecma Standard
35 posted drafts
ES5.1/ISO 16262:2011
2008
2009
2010
2011
2012
2013
March 11, IE9 ES5, except strict mode
March 22, FF4, full ES5
July, Safari 5.1, ES5, except bind method
August, Chrome 13, full ES5
Feb, Opera Mobile12, full ES5
Dec, Opera Desktop 11.60, full ES5
Oct., IOS 5, ES5 except bind
May-July?, Android Browser, ES5 except strict mode

ES5 World-Wide Browser Share January 2012

Desktop Browsers



Mobile Browsers



As a web developer, when will you be able to safely assume that all your users will be using an ES5 browser?

Based upon netmarketshare.com world-wide data:
http://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0
http://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=1

So why should you care about "ES Harmony"?

- You care about the long term evolution of JavaScript and the Web Platform.
- If you are a developer, you are probably going to be using JavaScript for much of your career

But don't look here to find immediately usable solutions to today's problems

How does TC-39 work?

It's not like this…

- TC-39 members participate in, listen to, and observe the web developer community.
- TC-39 reaches consensus on broad goals for future edition(s)
- Feature Champions prepare straw man proposals for wiki.ecmascript.org
- Discussed extensively on es-discuss and at F2F meetings
- Prototype implementations are encouraged, particularly in real browsers.

How does TC-39 work?

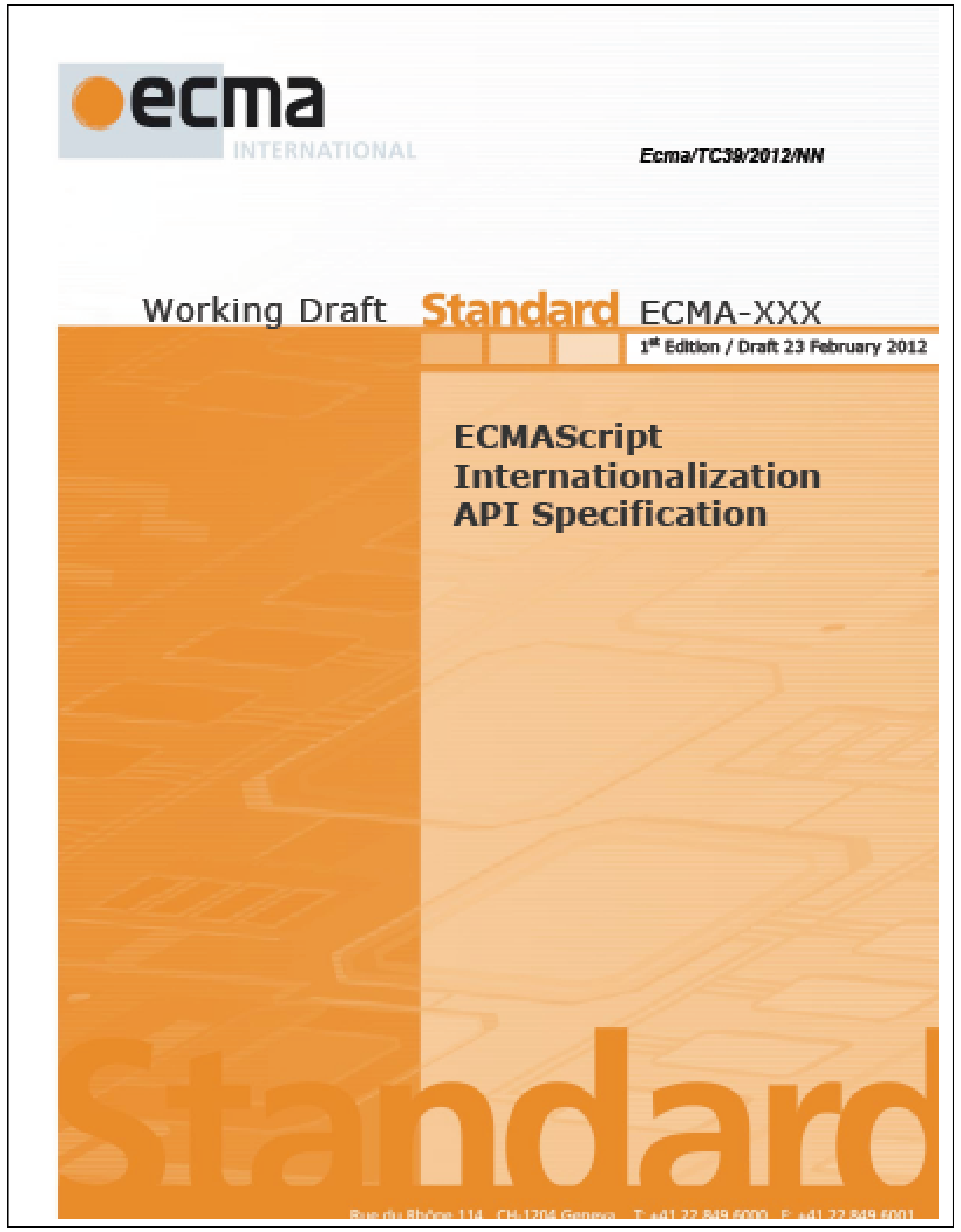
…and not like this, either.

- For each proposal consensus is reach to either drop, accept, or iterate.
- Editor integrates a comprehensive specification into the "ES.next" draft.
- Ideally production implementations and test suites are developed prior to publication of "ES.next" standard
- Specifications for discrete functional subsystems or libraries may be issued as separate Ecma standards.

ECMAScript Internationalization API

- Locale selection and multiple locale support
- Locale based:
  - ✓ String Collation
  - ✓ Number Formating
  - ✓ DateTime Formating
- Version 1 expected to be final this year (December 2012)

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



ecma INTERNATIONAL

Ecma/TC39/2012/NN

Working Draft Standard ECMA-XXX

1st Edition / Draft 23 February 2012

ECMAScript Internationalization API Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

ECMAScript Harmony Goals

1. Be a better language for writing:
    A. complex applications;
    B. libraries (including the DOM) shared by those applications;
    C. code generators targeting the new edition.
2. …

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

Things we are focusing on for ES.next

- Modularity
- Better Abstraction Capability
  - Better functional programming support
  - Better OO Support
- Expressiveness
- Things that nobody else can do

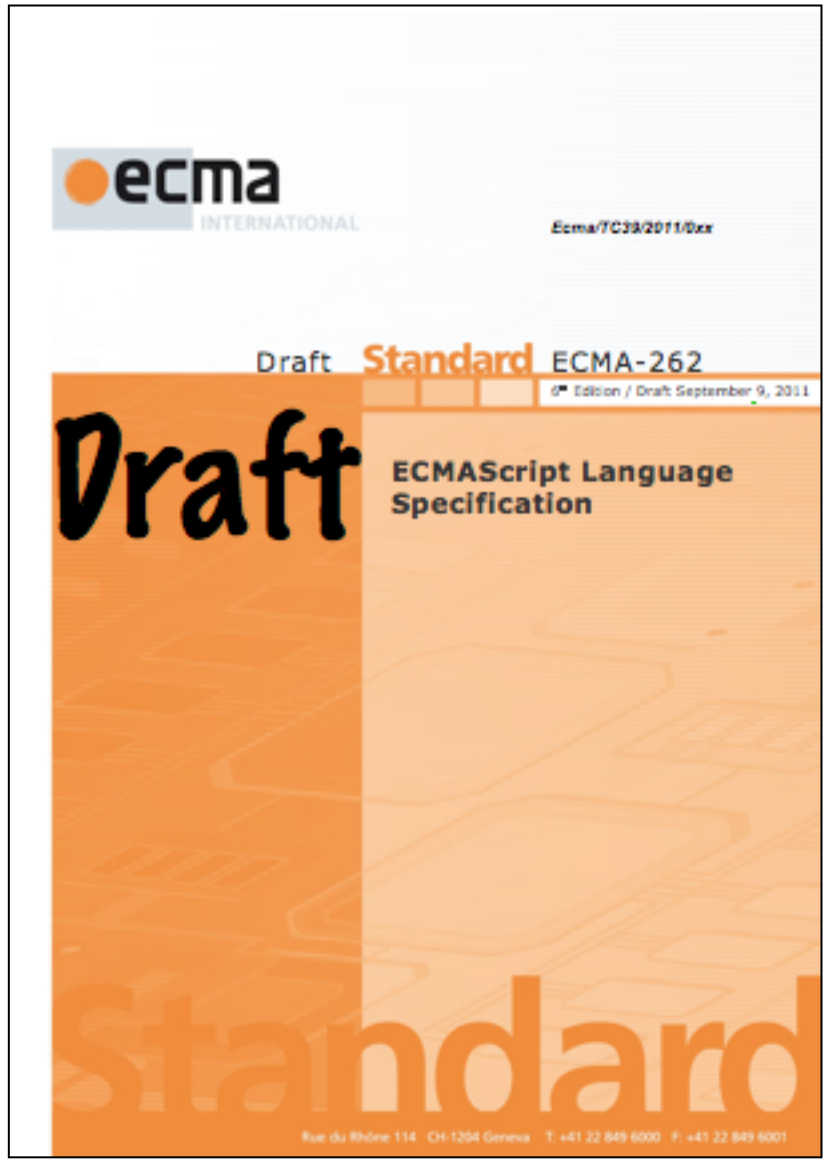
What Kind of Language Is JavaScript?

- Functional?
- Object-oriented?
  - Class-based?
  - Prototype-based?
- Permissive?
- Secure?

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

Some ES.next Enhancements

- Modules and Sanding-boxing module loaders
- Control abstraction via iterators and generators
- Array comprehensions
- String interpolation
- Binary Data Objects
- Built-in Hash maps and sets.
- More built-in Math and String functions



Ecma/TC39/2011/0xx

Draft Standard ECMA-262

6th Edition / Draft September 9, 2011

Draft

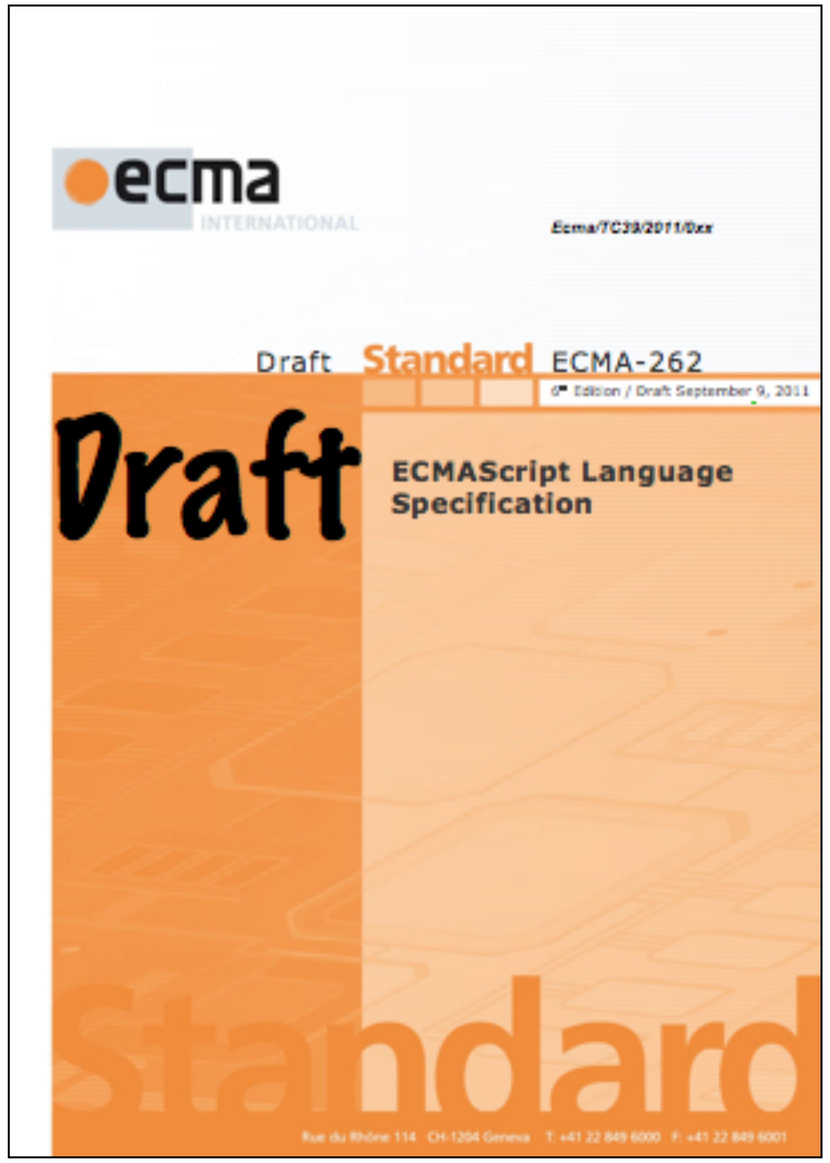
ECMAScript Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

More ES.next Enhancements

- super method calls references
- Encapsulated state via gensym-like private names
- More concise and powerful Object literal forms
- “Subclassable” built-ins, including Array
- Proxy Objects – low level behavioral intercession



ecma INTERNATIONAL

Ecma/TC39/2011/0xx

Draft Standard ECMA-262

6th Edition / Draft September 9, 2011

Draft

ECMAScript Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

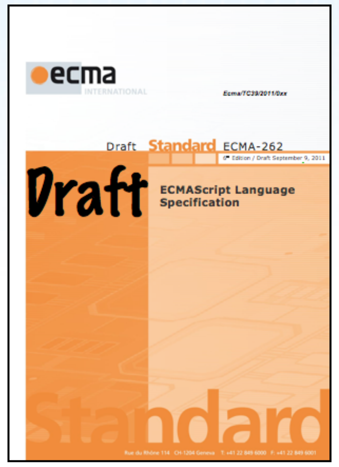
ES5 Ad Hoc Modularity

```
var collections = function(hashes) {
   function Dictionary() {
     ...
     var h=hashes.IdentityHash(obj);
     ...
   }
   ...
   return {
      Dictionary: Dictionary,
      Set: Set
   }
}(HashFunctions);
```

The Module Pattern

Modules

ES.next


ecma

Draft Standard ECMA-262

Draft

ECMAScript Language Specification

Standard

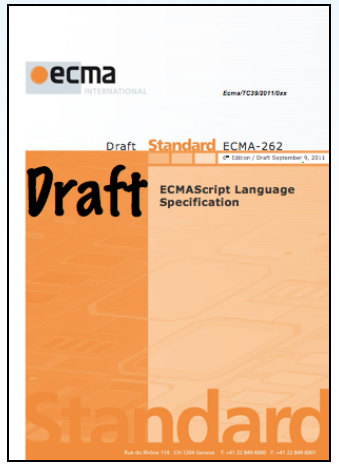
ES.next has syntactic modules

```
module Collections {
   import IdentifyHash from HashFunctions;
   export function Dictionary () {
     ...
     var h=IdentityHash(obj);
     ...
   }
   ...
}

import {Dict: Dictionary} from Collections;
Import $ from “jquery.js”;
```

Block Scoping
“let is the new var”

ES.next



```
for (let p of values(obj)) {
    let sinc = new Sinc;
    controls[p].onclick = function() {sinc(p)}
}
```

With temporal dead zones:

```
{
    function csq() {return c*c};
    let x = csq();
    const c=3;
    const k=csq();
}
```

Forward references to c within function csq

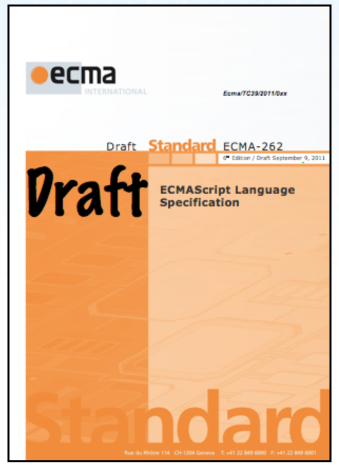
Throws because csq called before c is initialized

c is initialized

It now ok to call csq

Funtion Parameters and Destructuring

ES.next



Default value parameters, rest parameters, spread operator

```
function f(a, b=2) { };
function g(req, ...args) {
   foo(...args)
}
```

Optional parameter

Rest parameter

Spread operator

Destructuring in assignment, declarations, and formal parameters.

Object destructuring

```
const {x,y} = getPoint( );
let [first, ...rest] = someArray;

PointProto.add=function({x:argX=0,y:argY=0}){
   return new Point(this.x+argX, this.y+argY);
}
```

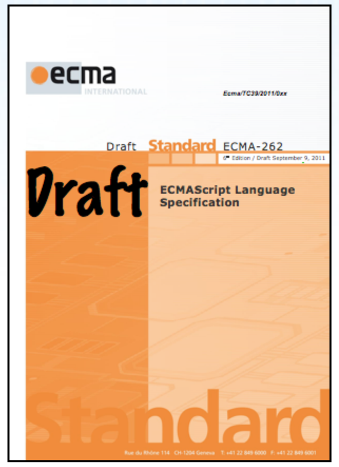
Array destructuring declaration

Destructuring parameter

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

Private Property Names

ES.next


ecma

Draft Standard ECMA-262

Draft

ECMAScript Language Specification

Standard

You can only access the property if you have access to the name object.

```
const secretX = Name.create();
```

Creates a unique unforgable value that can used as a property key.

```
function Point(x,y) {
   this[secretX]=x;
   this[secretY]=y;
   this.addPt = function(pt) {
      return new Point(this[secretX]+pt[secretX],
                       this[secretY]+pt[secretY]);
   }
}
```

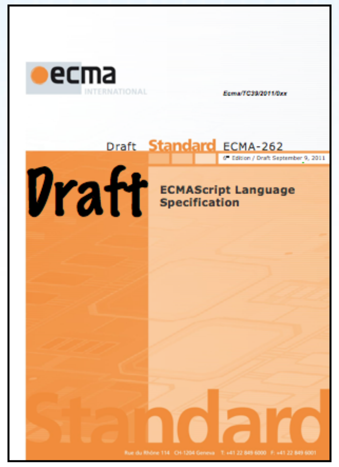
"Class" private instance variables

Also supports instance private, friend access, and other limited access patterns

Private Property Names

(or may be like this)

ES.next


ecma

Draft Standard ECMA-262

Draft

ECMAScript Language Specification

Standard

```
private secretX, secretY;
```

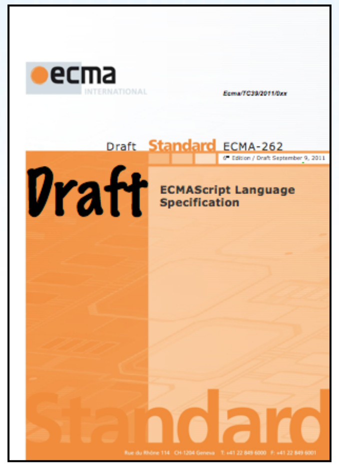
Creates a unique non-forgable value that can used as a property key.

```
function Point(x,y) {
   this.@secretX = x;   “Class” private instance variables
   this.@secretY = y;
   this.addPt = function(pt) {
      return new Point(this.@secretX+pt.@secretX,
                       this.@secretY+pt.@secretY);
   }
}
```

Check back next month…

More Concise & Expressive Object Literals

ES.next



```
function Point(x,y) {
    return Point.prototype <|{
        @secretX: x,
        @secretY: y,
        addPt(pt) {
            return new Point(this.@secretX+pt.@secretX,
                             this.@secretY+pt.@secretY)},
        toString() {
           return super.toString()+
                `(x:${this.@secretX} y:${this.@secretY})`}
      }
}
```

Set [[Prototype]] of Object literal

Private properties

Concise method property

super method call

String interpolation

What About Classes?

ES.next

```
class Point(x,y) {
      @secretX: x,
      @secretY: y,
      addPt(pt) {
         return new Point(this.@secretX+pt.@secretX,
                          this.@secretY+pt.@secretY)},
      toString() {
        return super.toString()+
            `(x:${this.@secretX} y:${this.@secretY})`}
    }
}
```

The Devil Is In the Details

ES.next Implementation Progess

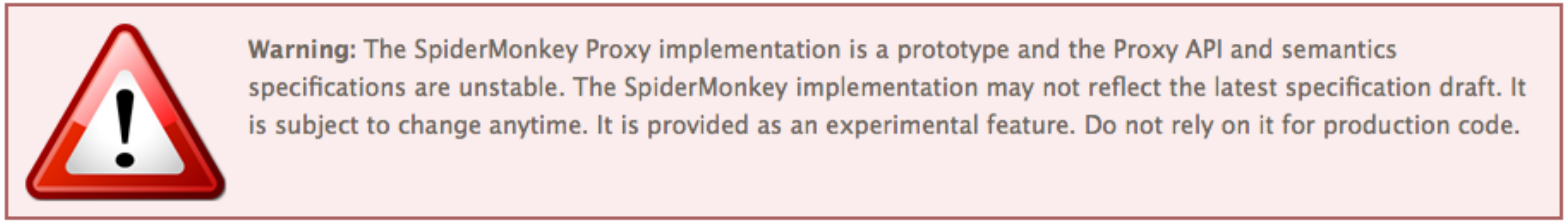
(March 2012)

FireFox

- ≈Block scoping/let/const (2006)
- ≈Destructuring (2006)
- ≈Iterators (2006)
- ≈Generators (2008)
- ≈Array Comprehensions (2008)
- ≤Weak Maps
- ≠Proxy

Chromium

- ≤Block scoping/let/const
- ≤Maps and Setsfff
- ≤Weak Maps
- ≠Proxy



Warning: The SpiderMonkey Proxy implementation is a prototype and the Proxy API and semantics specifications are unstable. The SpiderMonkey implementation may not reflect the latest specification draft. It is subject to change anytime. It is provided as an experimental feature. Do not rely on it for production code.

≈ Similar to ES.next
≠ Not current ES.next API
≤Current ES.next spec.

ECMAScript Resources

The Official ECMAScript 5.1 Specification (PDF)

- http://www.ecma-international.org/publications/standards/Ecma-262.htm

The Unofficial Annotated ECMAScript 5.1 Specification (HTML)

- http://es5.github.com/

Test262: The Offical ECMAScript Implementation Test Suite

- http://test262.ecmascript.org/

The TC-39 Wiki

- http://wiki.ecmascript.org

The TC-39 ECMAScript Design Discussion Mail List

- https://mail.mozilla.org/listinfo/es-discuss

“ES6” Specification Drafts

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

Please report bugs

- http://bugs.ecmascript.org

We're all collectively creating a new era of computing.

Enjoy it!