

“Post-PC Computing” Is Not a Vision

Talk, by [Allen Wirfs-Brock](#)

Strange Loop 2011, September 20, 2011

It's increasing obvious that we are in the midst of a transformation of computing that will be as significant and disruptive as the emergence of personal computing was thirty years ago. There are technical advancements that enable this change, just like there were for personal computing. But this time it seems less clear where the changes are taking us. The PC era wasn't just the result of random-walk application of emerging technologies. It was a revolution that was shaped by a shared vision. Where's our vision? Is there a goal that is shaping the current revolution? Is it really just about mobility, clouds, HTML, content, and “not PCs”? Change is roaring at us and that makes this a great time for innovative technologists. But let's find our shared vision and see if we can be intentional about using our innovations to shape the next era of computing.

mozilla

“Post-PC Computing” Is Not a Vision

Allen Wirfs-Brock

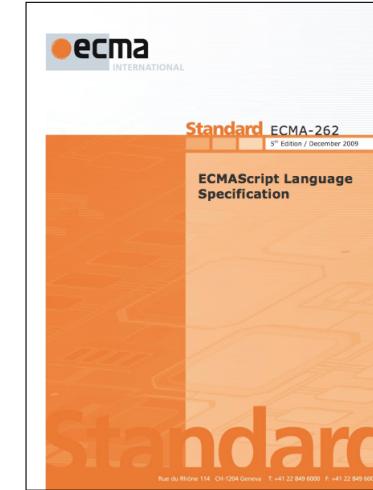
mozilla

A little background about me

- Compilers, Smalltalk virtual machines, GCs, language design, development tools
- Launched first commercial Smalltalk: Tektronix 4404
- Digitalk/Parcplace-Digitalk: Enterprise Scale Smalltalk
- (Re-) Instantiations: JOVE Java optimizing compiler, Eclipse tools
- Microsoft: JavaScript/ECMAScript 5
- Mozilla: Future of JavaScript and the Web platform



<http://youtu.be/8yxCIafayW-8>



mozilla

TODAY @ PCWORLD

Are We Really Living in a Post-PC World?

By Jason Cross, PCWorld Mar 4, 2011 1:30 PM



Among the many ear-catching bon mots issued forth from Steve Jobs recently is the assertion that we're living in, or at the very least *entering*, the post-PC era. It started last year at the [D8 conference](#), when the Apple CEO said that PCs are going to be "like trucks" in that they'll still be around and useful for certain work, but only a smaller percentage of the users will need one. More recently at the [iPad 2](#) launch, Jobs re-confirmed this post-PC mantra.

Apple's third post-PC blockbuster



broadcast

FILED UNDER: BUSINESS TECH

VMware boss focuses on post-PC era at VMworld

By Jay Greene
AUGUST 29, 2011 6:41 PM PDT



Print E-mail

Post-PC

iCloud to be the corner stone for Apple's post-PC era

By Édouard PLANTE | Culture, Social

July 5th, 2011

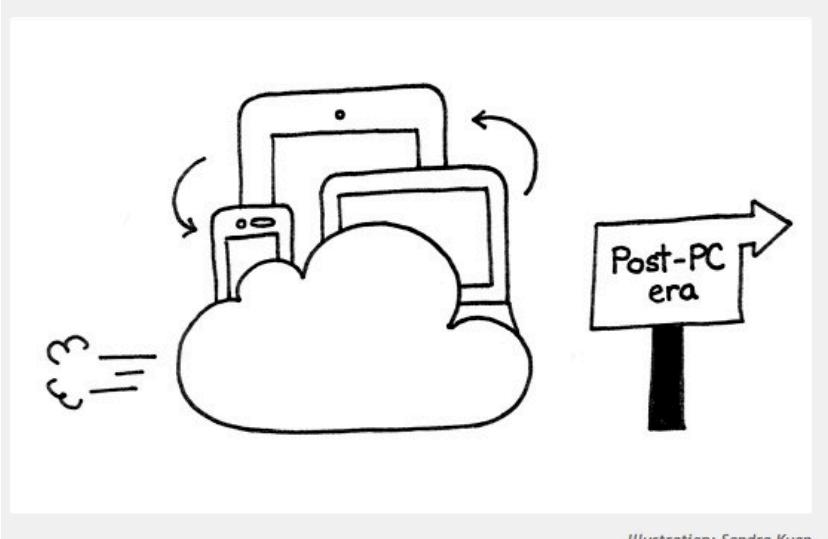


Illustration: Sandra Kuan

Remembering the Post-Mainframe Computing Era?

Introducing Apple II.

The home computer that's ready to work, play and grow with you.

Clear the kitchen table. Bring in the color TV. Plug in your new Apple II[™] and connect any standard cassette recorder/player. Now you're ready for an evening of discovery in the kitchen or the family room.

Only Apple II makes it that easy. It's a complete computer system. At \$1298, it includes features you won't find on other personal computers costing twice as much.

Features such as video graphics in 15 colors. And a built-in memory capacity of 8K bytes ROM and 4K bytes RAM—with room for lots more. And a built-in cassette interface so you can swap programs from audio cassettes, using the built-in cassette interface, so you can swap with other Apple II[™] computers or optional equipment on most personal computers, at hundreds of dollars extra and thousands of dollars less. All designed to keep up with changing technology, to expand easily whenever you need it.

An investment that's a wise financial investment. You can program it to tutor your children in most any subject, such as spelling.

But the biggest benefit—no matter how you use Apple II—is that you and your family increase your familiarity with the computer itself. The more you experiment with it, the more you'll realize its potential.

Start playing PONG. Then invent your own games using the input keyboard, game paddles and built-in speaker. As you experiment with the computer, you'll learn new skills which will open up new ways to use your Apple II. You'll learn to "paint" dazzling color graphics on the screen, to write programs in BASIC in ROM, and write programs to create beautiful kaleidoscopic designs. And you'll learn to use the computer to be able to organize, index and store data on household files—recipes, grocery lists, bills, checkbooks, recipes, and record collections. You can learn to chart your investments, balance your checking account, even control your home environment—all the things you can imagine far as your imagination can take it.

Best of all, Apple II is designed to grow with you. As you become more involved with computing experience, you may want to add new Apple peripherals. For example, a remote terminal board for connecting to another computer; a monitor for displaying what's being developed for advanced scientific and mathematical applications; a printer for printing out programs and data; and eight plug-in options such as a protecting board for experiments, a serial board for connecting telephone, printer and other terminals; a parallel interface for connecting to another computer; an EPROM board for storing programs permanently; and a modem for connecting to other computers via telephone line. A complete disk interface with software and complete operating systems will be available at the end of the year. All designed to accommodate you, because Apple II was designed from the beginning to accommodate increased requirements as your requirements change.

If you'd like to see for yourself how easy it is to use and enjoy Apple II, call 408/253-1000 for a demonstration and a copy of our detailed brochure. Or write Apple Computer Inc., 1680 Amphitheatre Parkway, Cupertino, California 95014.

apple computer inc.

Apple II[™] is a completely self-contained computer system with BASIC in ROM, color graphics, ASCII keyboard, lightweight, efficient switching power supply and built-in cassette interface. BASIC in ROM, up to 48K bytes of RAM, and with cassette tape, video and disk drives. Other features included are two game paddles and a demonstration cassette.

SPECIFICATIONS

- **Memory:** 8K bytes (16K total); RAM: up to 48K bytes on-board
- **Video Display:** Memory mapped, 5 modes—all Software selectable
- **Character:** 16 characters wide, 24 lines upper case
- **Color graphics:** 409 x 404, 16 colors
- **Keyboard:** 50 keys, including numeric keypad, 102x, black, white, violet, green (16K RAM minimum required)
- **Power:** 120VAC, 60Hz, 150W, controlled to include 4 lines of text at the bottom of the display area
- **Ports:** One for permanent memory access. All color generation done digital
- **Memory:** up to 48K bytes on-board
- **RAM:** 4K supplied
- **Character:** 16K or new 16K dynamic memory chips
- **Up to 128K RAM (8K supplied)**
- **Video:** 16 colors
- **Fast extended Integer BASIC in ROM with color graphics commands**
- **Monitor:** Monochrome monitor in ROM
- **I/O:**
 - 1500 bps cassette interface
 - RS-232C port
 - Apple game I/O connector
 - ASCII keyboard port
 - Speaker
 - Composite video output

Apple II is also available in board-only configuration for the do-it-yourself hobbyist. Has all of the features of the Apple II system, but does not include case, keyboard, power cord or game paddle. \$398.

PONG is a trademark of Atari Inc.

*Apple II plugs into any standard TV using an inexpensive modulator (not supplied).

mozilla

Post-Post-Mainframe Era?

mozilla

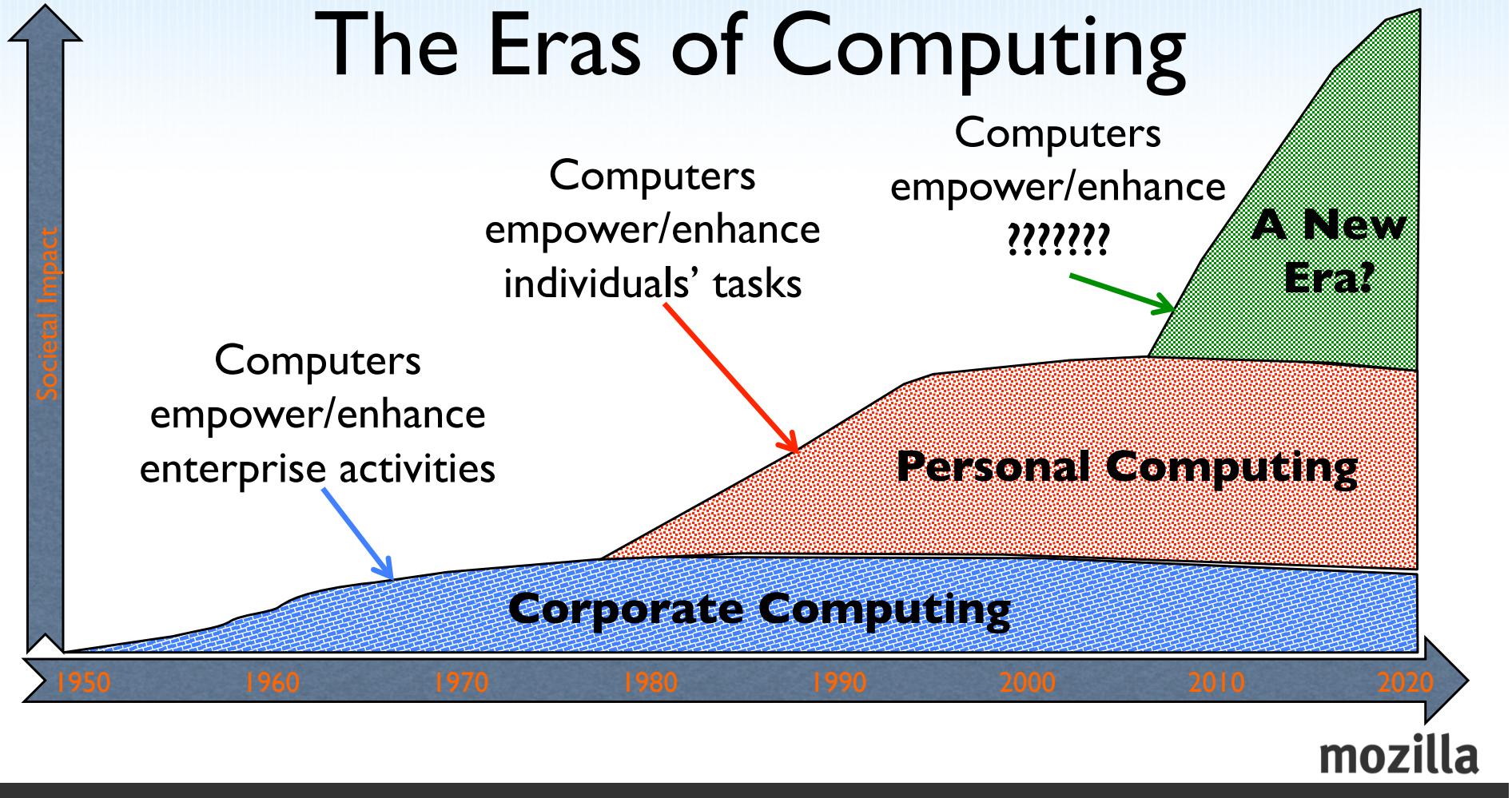
Hardware evolution doesn't define a computing era



Computing eras are about the impact of computing on society

mozilla

The Eras of Computing



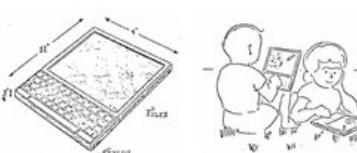
The PC Era was built around a revolutionary vision



n|w Fachhochschule Nordwestschweiz
Pädagogische Hochschule

Alan Kay: A Personal Computer for Children of All Ages

Xerox Palo Alto Research Center 1972



puter viewed as a medium can simulate any other medium if the means of simulation are sufficiently well described. Moreover, unlike conventional media, which are passive in the sense that marks on paper, paint on canvas, and television images do not change

COMPUTER SIMULATION'S general & Sutherland Computer Corporation is possible to present on a compact personal-computer system programs that revise an in-space objects in project their own National Aeronautics and Space Administration the interior of the space shuttle. The sequence, shows the movement of computer to simulate real or imagined phenomena.

EXPERIMENTAL PERSONAL COMPUTER was built at the Xerox Palo Alto Research Center as part of a high-level programming language that would enable nonexperts to write sophisticated programs. The machine is completely self-contained, consisting of a keyboard, a pointing device, a high-resolution picture display and a sound system, all connected to a small processing unit and a removable solid-state memory. Display can refresh thousands

Microelectronics and the Personal Computer

Rates of progress in microelectronics suggest that in about a decade many people will possess a notebook-size computer with the capacity of a large computer of today. What might such a system do for them?

by Alan C. Kay

response to the viewer's wishes, the computer medium is active; it can respond to queries and experiments and can engage the user in a two-way conversation.

The evolution of the personal computer has followed a path similar to that of the printed book, but in 40 years rather than 600. Like the handmade books of the Middle Ages, the massive com-

and organize a large quantity of information about their patients, enabling them to perceive significant relationships that were previously hidden. Composers should be able to hear a composition as they are composing it, notably if it is too complex for them to play. Businessmen should have an active briefcase that contains a working simulation of their company. Educators



If we are entering a new era of computing, what is the vision?

- The Mobile Era?
- The Cloud Era?
- The Tablet Era?
- The Games and Media Player Era?
- The Monetize the User Era?

mozilla

The Ambient Computing Era

- Devices not Computers
- Ubiquitous access to information
- Computer 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!

mozilla

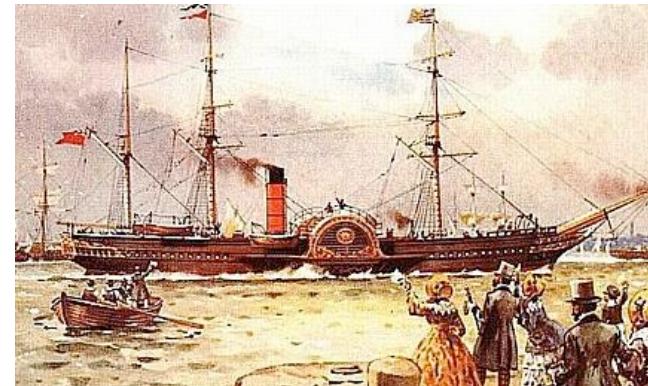
Other Common Labels

- Pervasive
- Ubiquitous
- Ambient Intelligence
- The Internet of Things

mozilla

Transitional Technologies

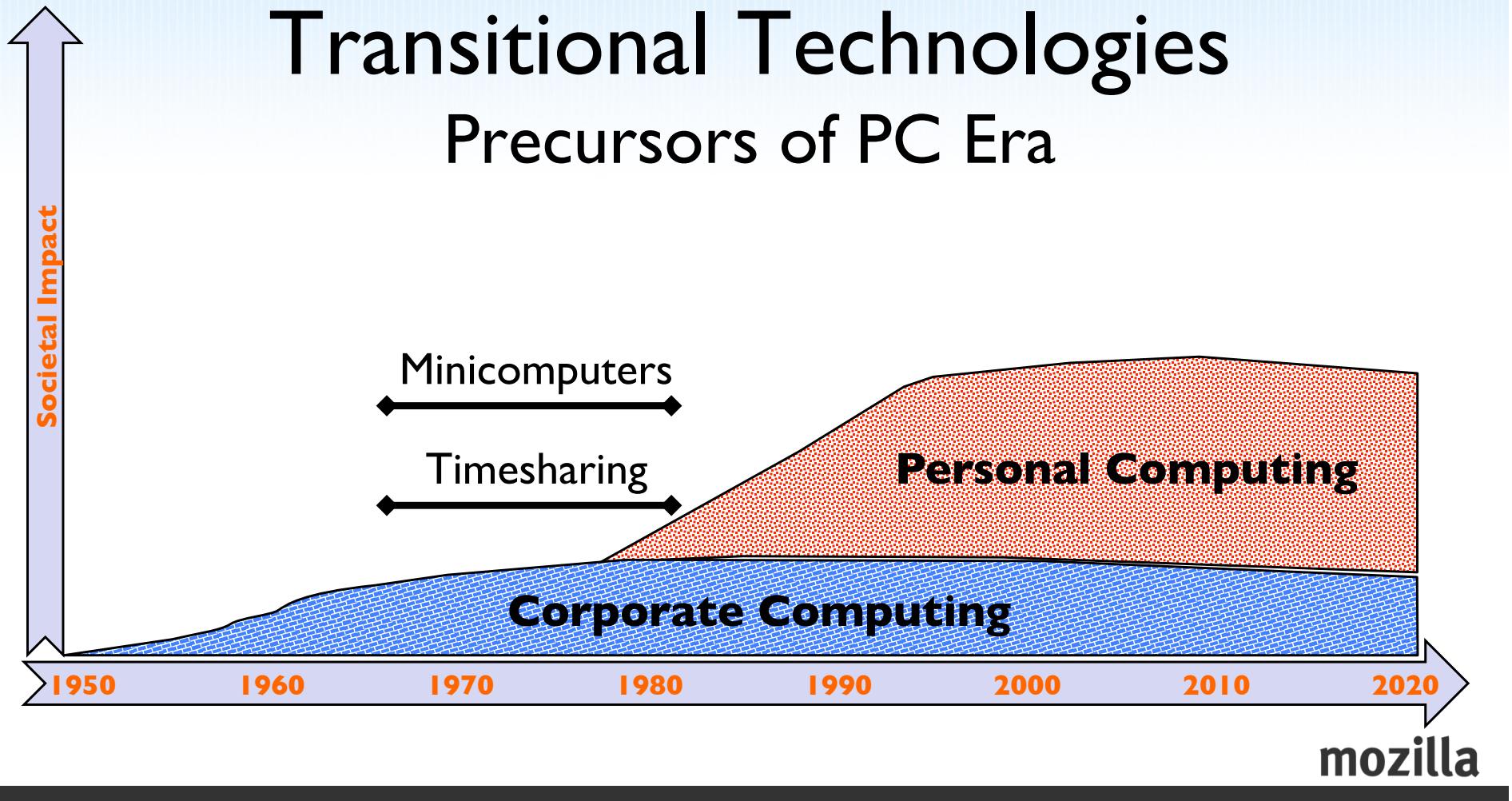
- Precursors of fundamental change
- Exhibit some characteristics of what will come
- Firmly root in the status quo



mozilla

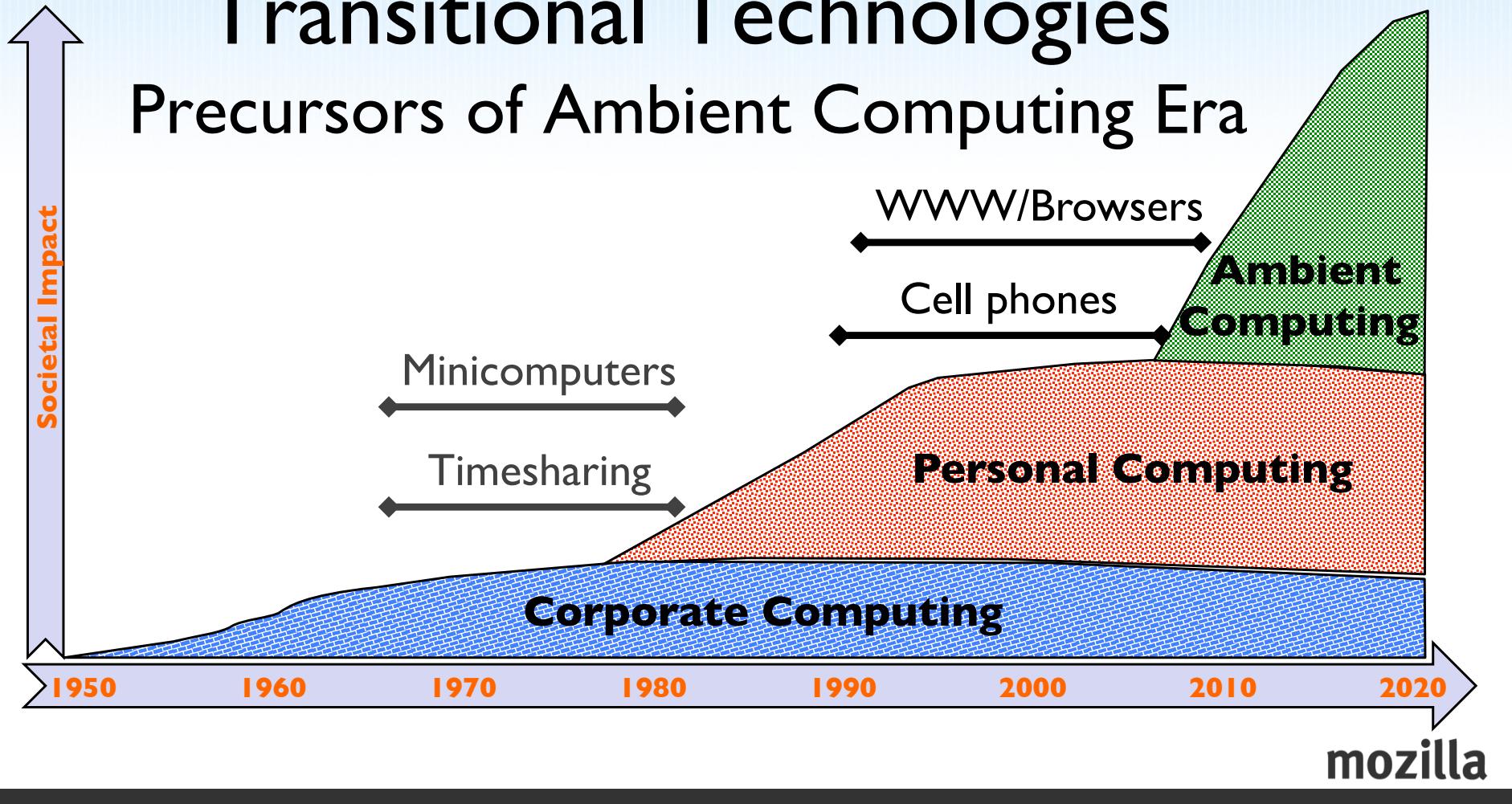
Transitional Technologies

Precursors of PC Era



Transitional Technologies

Precursors of Ambient Computing Era



The Cell Phone

- Always with you
- Always connected
- But initially dedicated into a single use case
- Legacy business models



mozilla

The WWW/Web Browser



- First real ubiquitous web of information
- Universal client/server computation model (HTTP/REST)
- Universal presentation platform
- Availability more important than performance
- A Web Browser is a classic PC application

mozilla

Why are these only transitional

- Telephony is just an application
- Wireless communications now common part of most devices
- WWW data access is an essential part of most applications
- Users don't "go to a pc" to use the web any more
- Growing focus on solutions (apps) not tools (the browser)
- When something becomes ubiquitous, it disappears

mozilla

Every Computing Era Has a Dominant Application Platform

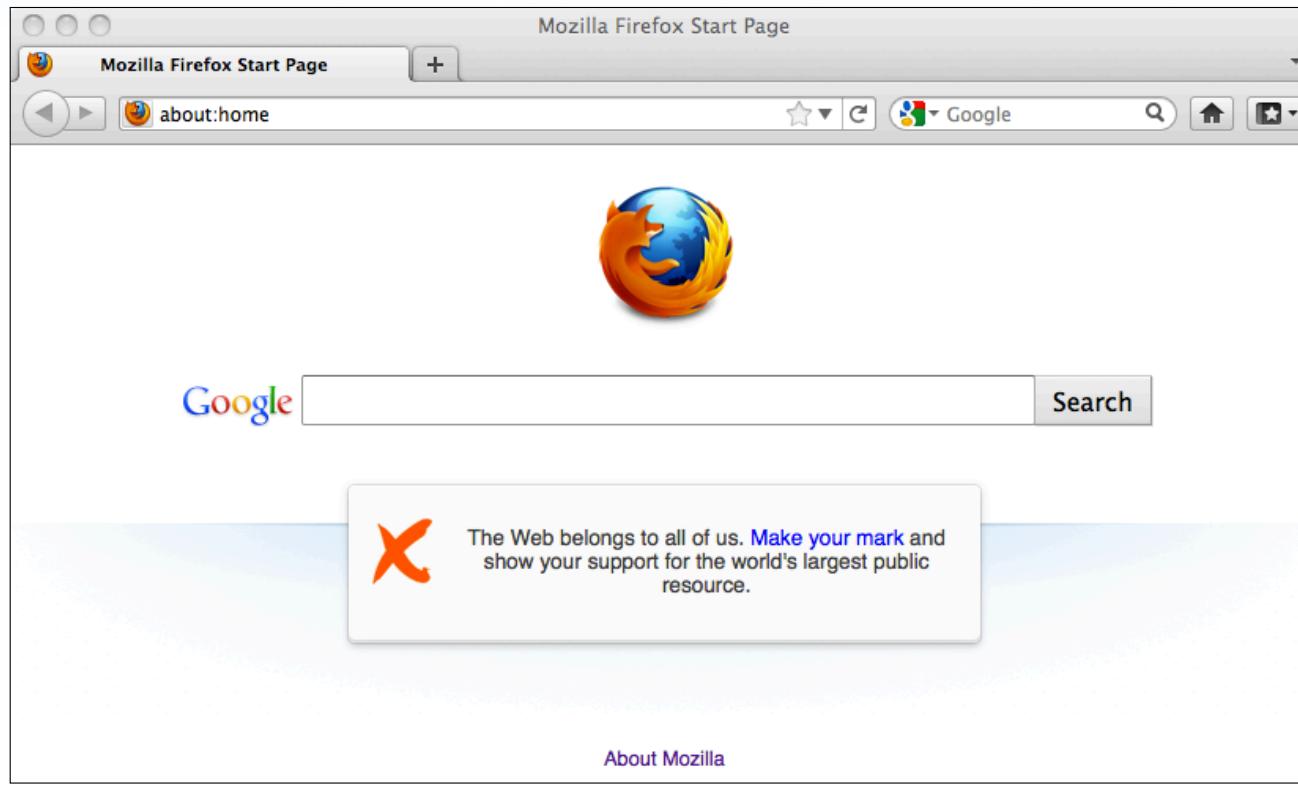
- Corporate Computing Era: IBM Mainframes
- Personal Computing Era: Microsoft/Intel PC
- Ambient Computing Era: T.B.D (or is it?)

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

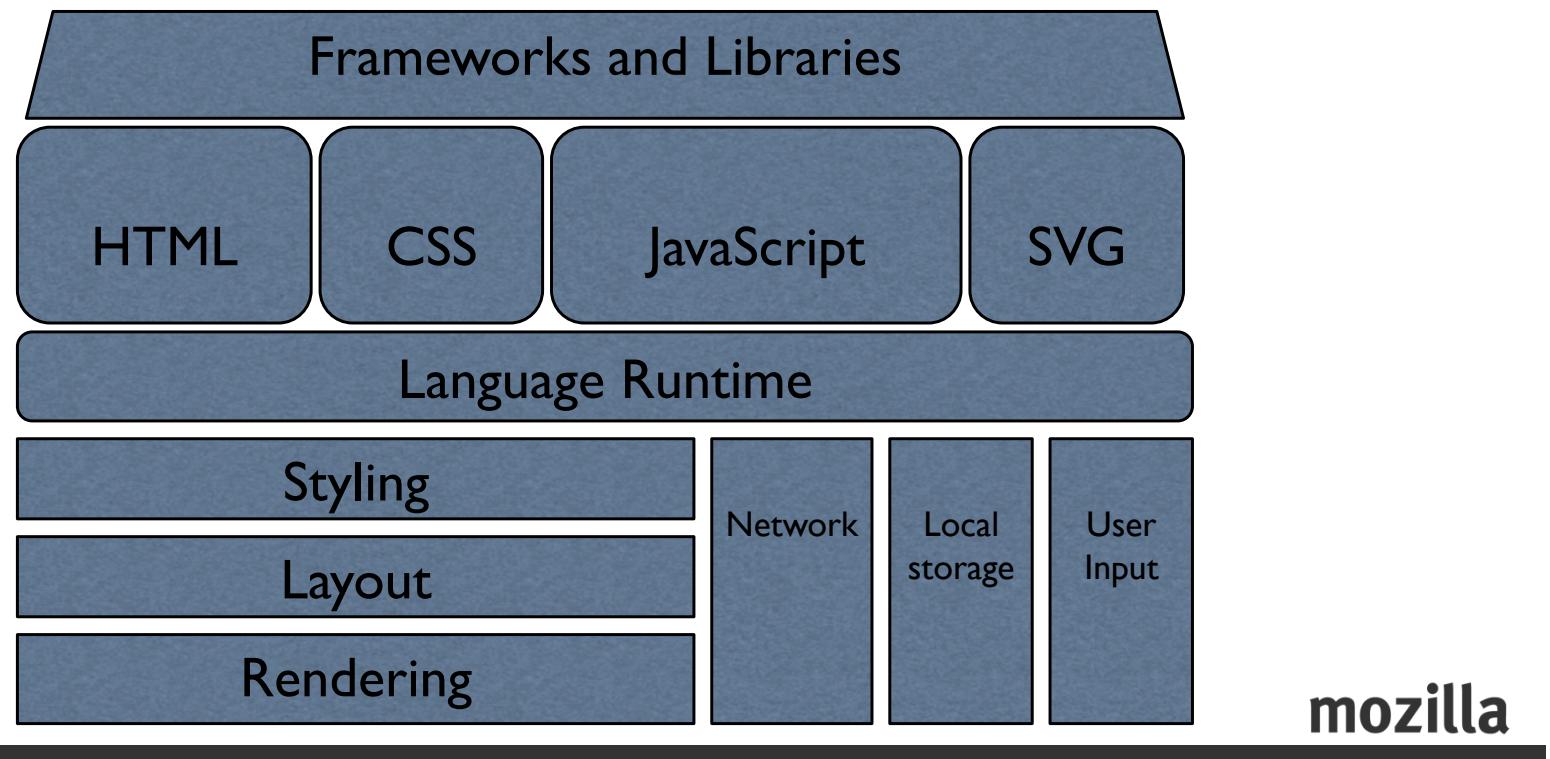


mozilla

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



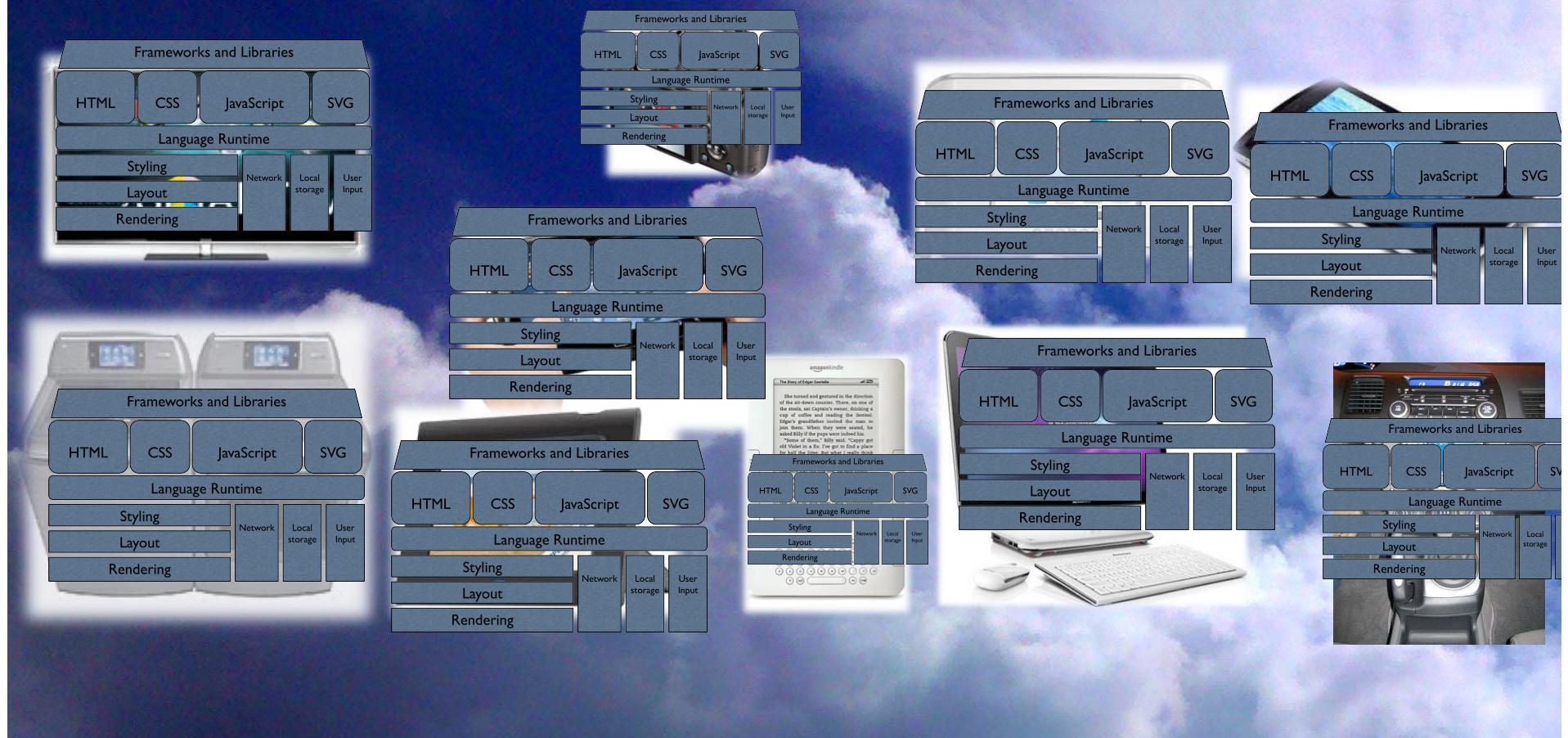
What do you have when you strip away the PC application part of a web browser?



The Ambient Applications Platform



The Ambient Applications Platform

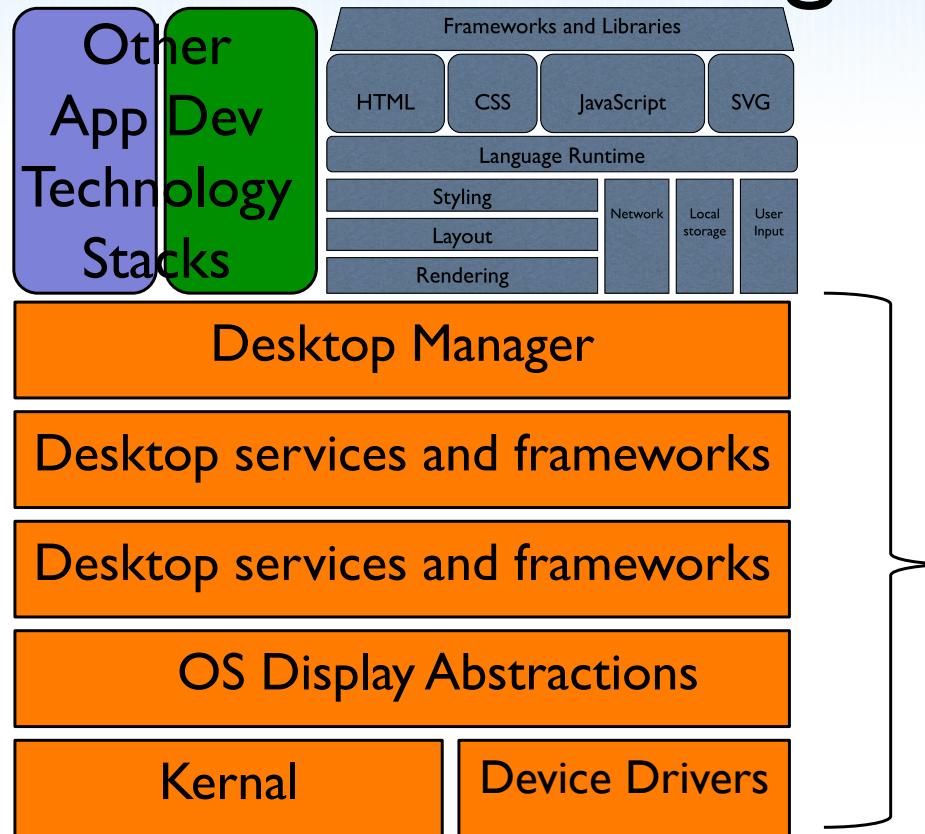


Web Apps vs. “Apps”

- HTML/CSS/JavaScript/WebApp APIs
- Server via: HTTP, JSON, XML, Sockets
- Works on any device
- Task specific “Chromeless” UIs emerging
- “App Stores” about to emerge
- Proprietary Language+Frameworks (Java, Objective-C, C#, etc.)
- Server via: HTTP, JSON, XML, Sockets
- Works only within a device family
- Task specific “Chromeless” UIs
- Tied to a specific “App Store”

mozilla

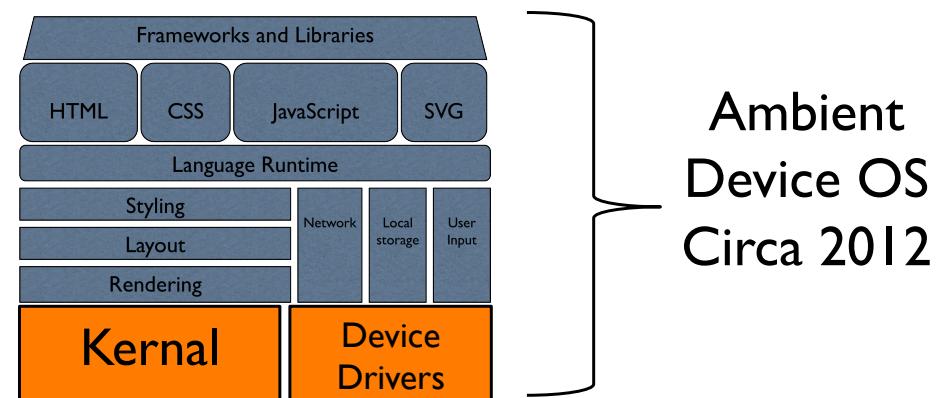
New Platform Emergence



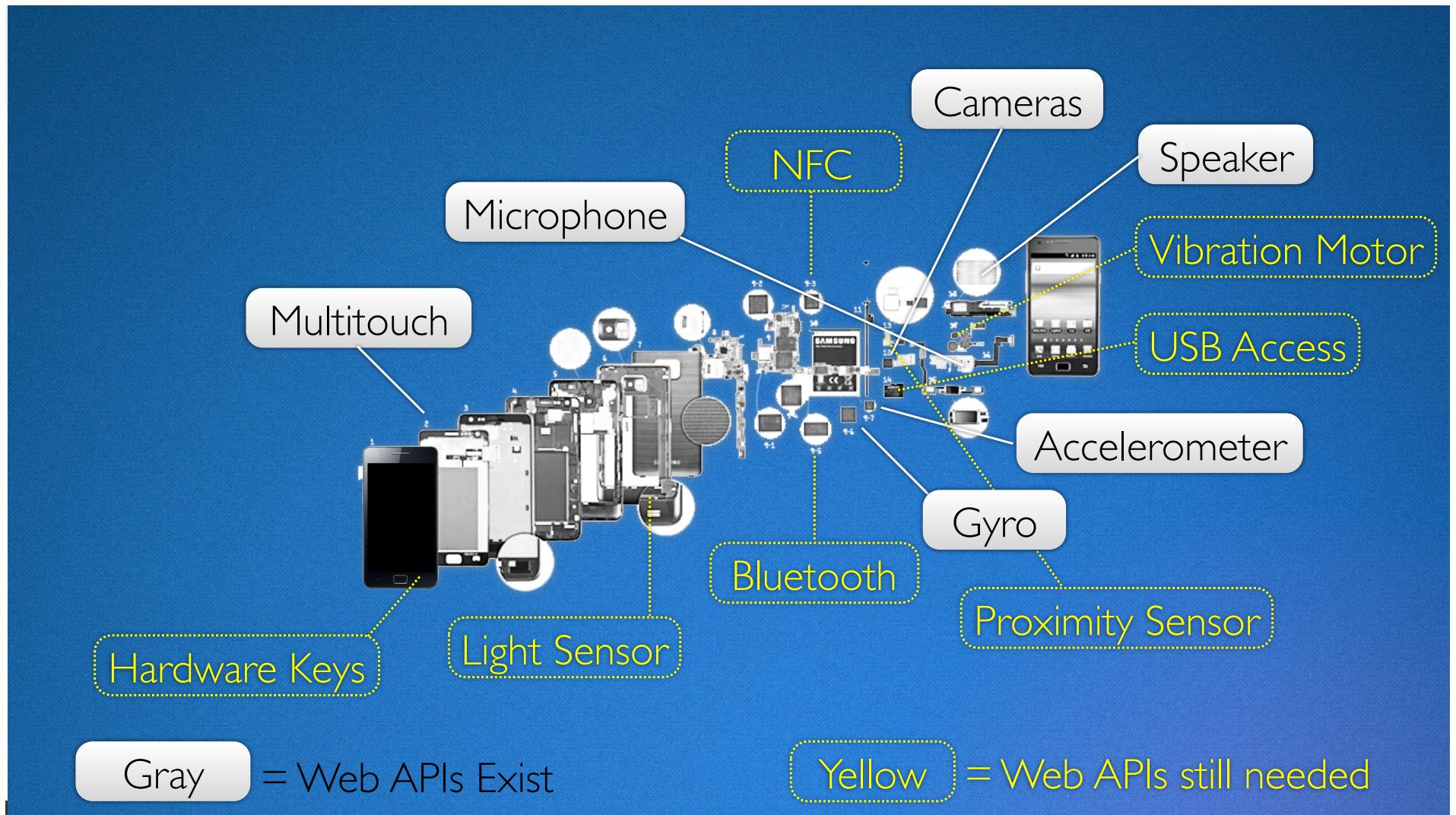
mozilla

New Platform Emergence

- Mozilla Boot to Gecko
- Chrome OS
- Plam/HP WebOS
- Etc.



mozilla

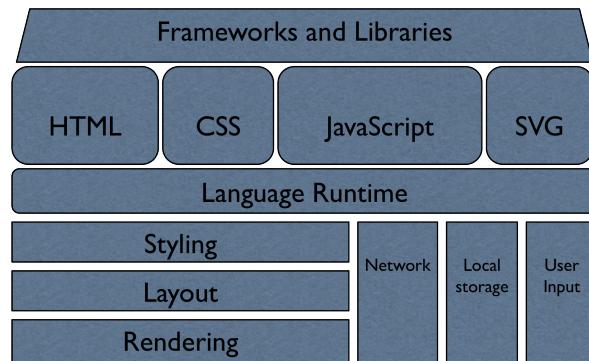


Proprietary Silos



Walled Gardens

or a
Standards-based
Open Platform?



mozilla

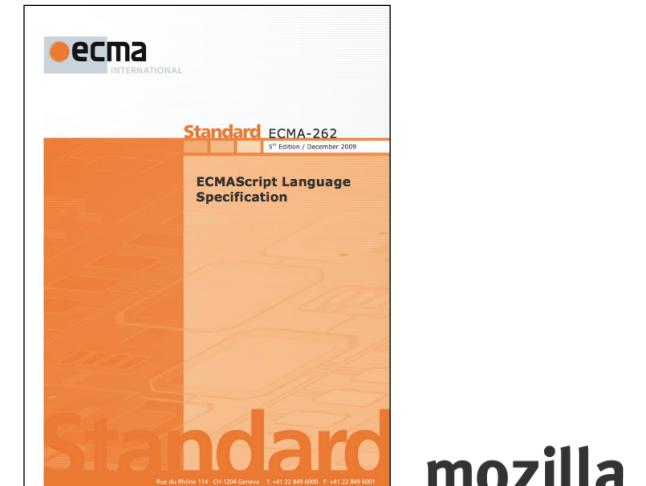
Areas of real concern

- Is it possible for a vendor neutral standards-based open platform to achieve similar dominance?
- Cumbersome standardization processes
 - Really! W3C, Ecma, WhatWG, IETF, Kronos, ...
- Who really drives innovation?
- Slow rate of change
- Who provides the implementations?

mozilla

Each Computing Era has had a Canonical Programming Language

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



Why JavaScript?

Because “Worse is Better”

Dick Gabriel
<http://www.dreamsongs.com/WorselsBetter.html>

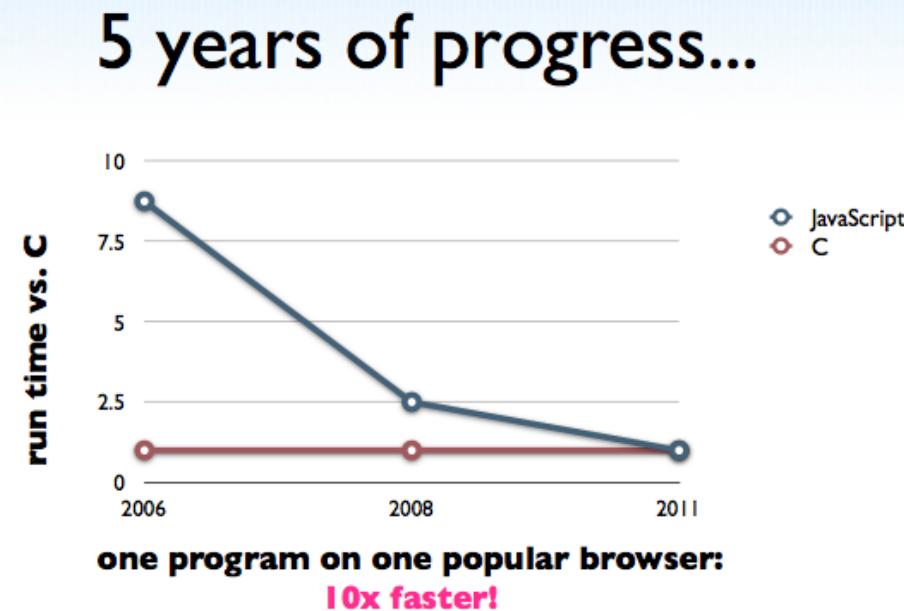
- It's there – It's working
- It's good enough
- It's getting better
- What could replace it?
- How could that happen?



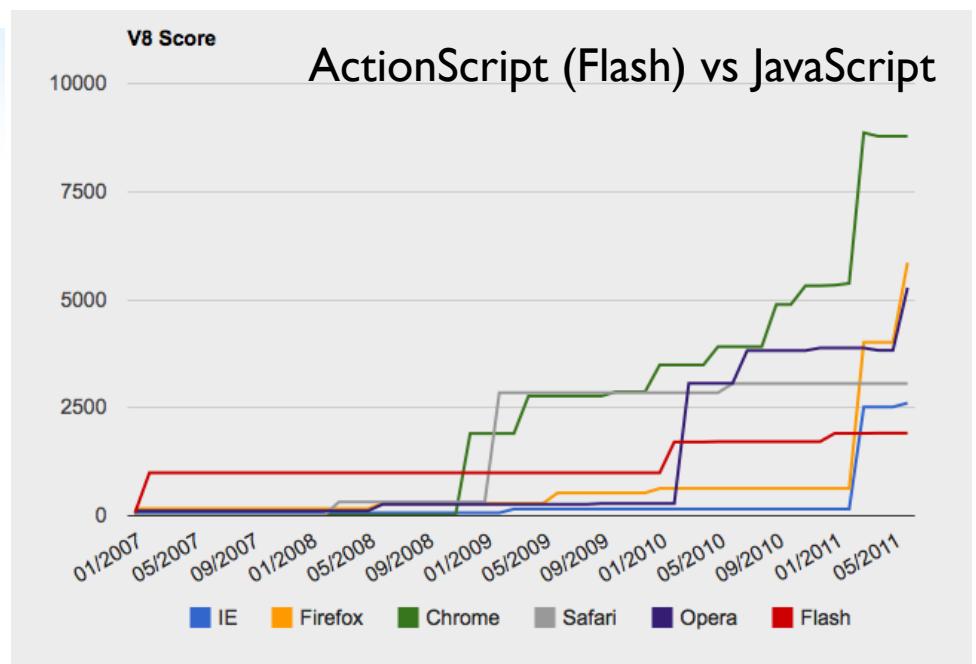
<http://odetocode.com/Blogs/scott/archive/2009/03/18/signs-that-your-javascript-skills-need-updating.aspx>

mozilla

JavaScript Performance Over Time



http://people.mozilla.com/~dmandelin/KnowYourEngines_Velocity2011.pdf



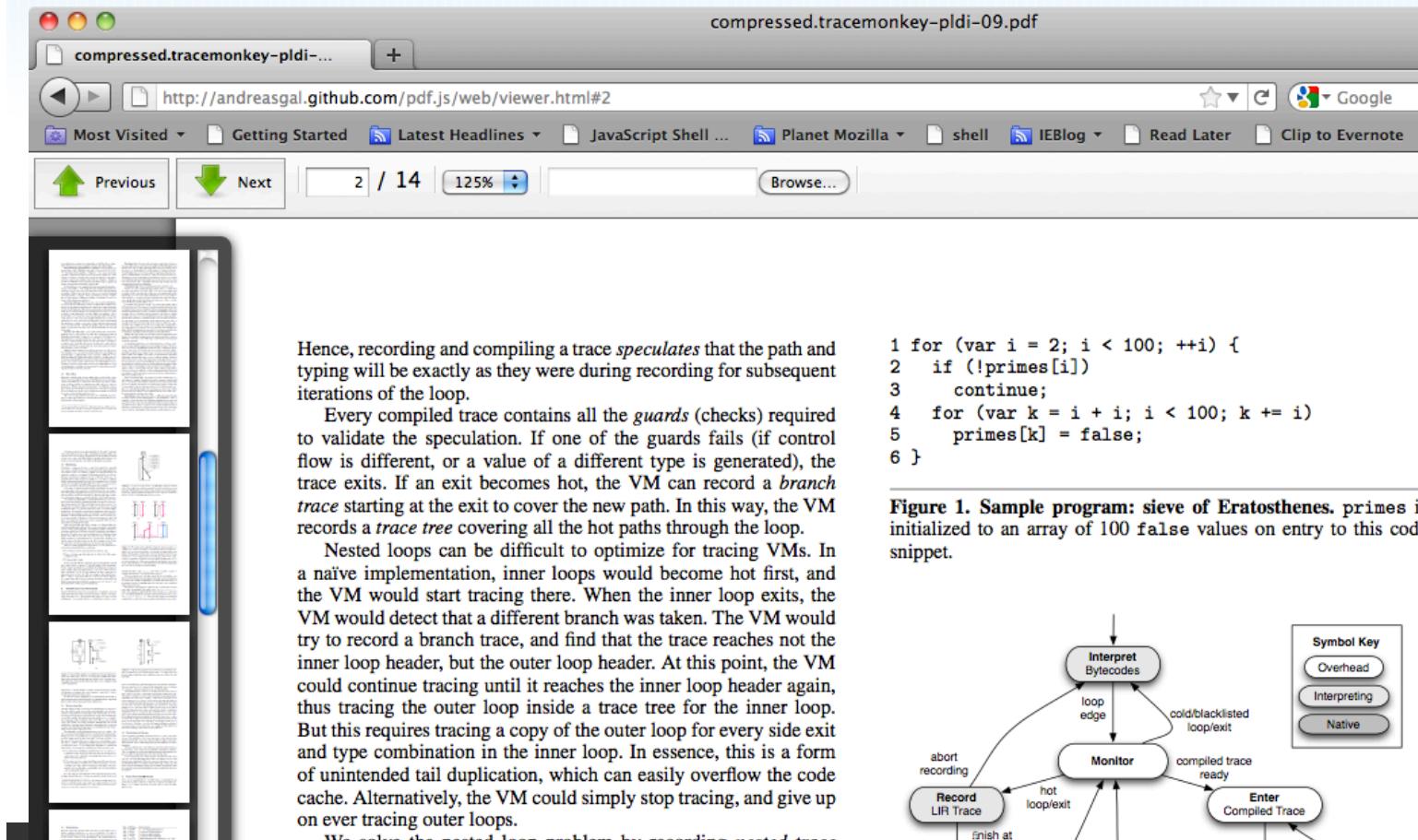
<http://iq12.com/blog/as3-benchmark/>

mozilla

PDF Renderer in JavaScript

<http://andreasgal.github.com/pdf.js/web/viewer.html>

<https://github.com/andreasgal/pdf.js>



The screenshot shows a web browser window with the title bar 'compressed.tracemonkey-pldi-09.pdf'. The address bar contains the URL 'http://andreasgal.github.com/pdf.js/web/viewer.html#2'. The browser interface includes standard controls like back, forward, and search, along with tabs for 'Most Visited', 'Getting Started', 'Latest Headlines', 'JavaScript Shell ...', 'Planet Mozilla', 'shell', 'IEBlog', 'Read Later', and 'Clip to Evernote'. Below the address bar, there are buttons for 'Previous' and 'Next', a page number '2 / 14', a zoom level '125%', and a 'Browse...' button. The main content area displays the PDF document.

Hence, recording and compiling a trace *speculates* that the path and typing will be exactly as they were during recording for subsequent iterations of the loop.

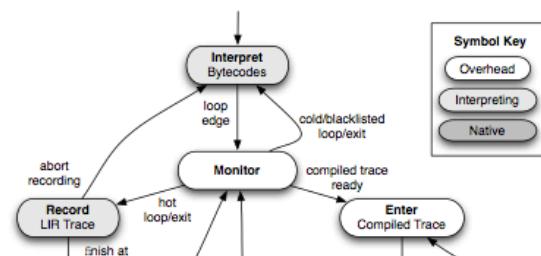
Every compiled trace contains all the *guards* (checks) required to validate the speculation. If one of the guards fails (if control flow is different, or a value of a different type is generated), the trace exits. If an exit becomes hot, the VM can record a *branch trace* starting at the exit to cover the new path. In this way, the VM records a *trace tree* covering all the hot paths through the loop.

Nested loops can be difficult to optimize for tracing VMs. In a naive implementation, inner loops would become hot first, and the VM would start tracing there. When the inner loop exits, the VM would detect that a different branch was taken. The VM would try to record a branch trace, and find that the trace reaches not the inner loop header, but the outer loop header. At this point, the VM could continue tracing until it reaches the inner loop header again, thus tracing the outer loop inside a trace tree for the inner loop. But this requires tracing a copy of the outer loop for every side exit and type combination in the inner loop. In essence, this is a form of unintended tail duplication, which can easily overflow the code cache. Alternatively, the VM could simply stop tracing, and give up on ever tracing outer loops.

We solve the nested loop problem by recording *nested trace*.

```
1 for (var i = 2; i < 100; ++i) {
2   if (!primes[i])
3     continue;
4   for (var k = i + i; i < 100; k += i)
5     primes[k] = false;
6 }
```

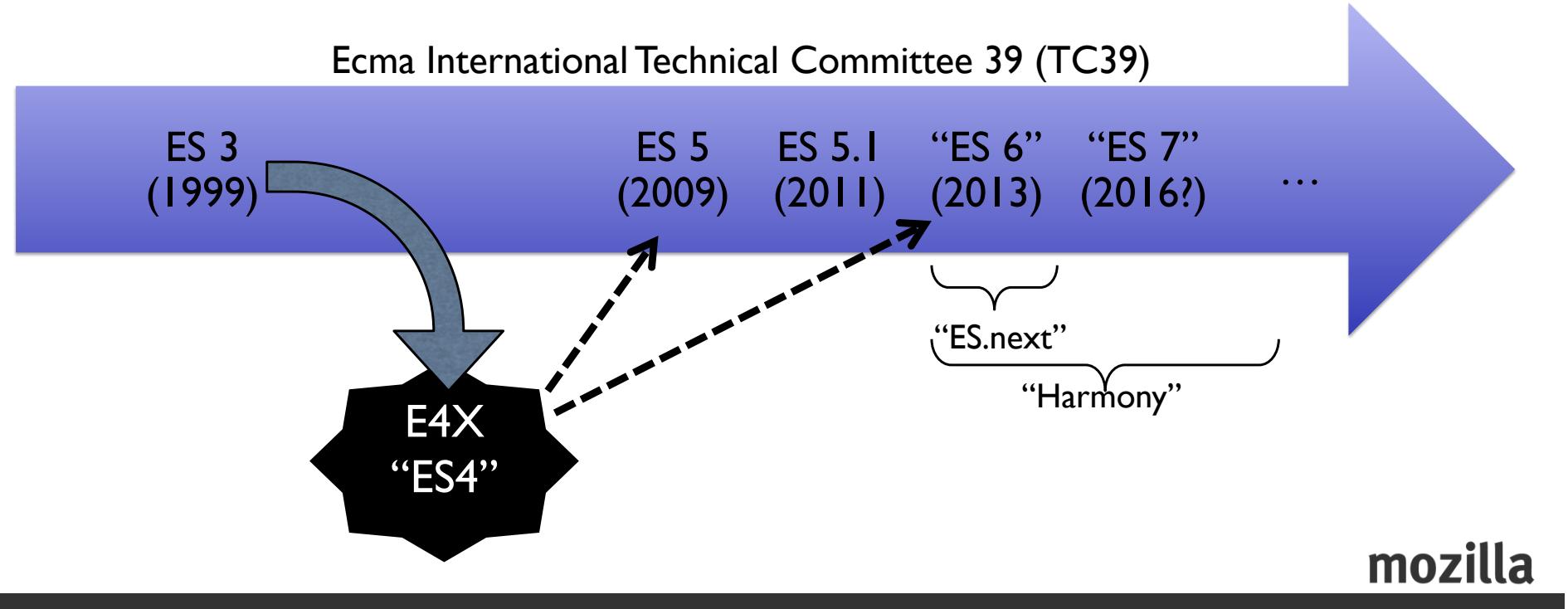
Figure 1. Sample program: sieve of Eratosthenes. `primes` is initialized to an array of 100 false values on entry to this code snippet.



mozilla

Evolving JavaScript

The ECMAScript Standards Process



Evolving JavaScript

The ECMAScript Standards Process

Ecma International Technical Committee 39 (TC39)

ES 3
(1999)

ES 5 ES 5.1 “ES 6” “ES 7”
(2009) (2011) (2013) (2016?) ...

- Classic ISO targeted “pay to play” standards organization
- Open public discussion list es-discuss@mozilla.org
- Working documents all publically available at wiki.ecmascript.org

“ES.next”
“Harmony”

mozilla

Interoperability is TC39's highest priority

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

ecma^{script} 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.

mozilla

Evolving JavaScript ECMAScript Harmony Goals

- I. 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>

mozilla

Amber Smalltalk

<http://nicolaspetton.github.com/amber/index.html>

Fork me on GitHub

Overview · Download · Learn · Documentation · Source

Amber
Smalltalk brought to the web

Transcript Workspace SUnit x Browser: Browse... +

Benchfib
Canvas
Compiler
Examples
IDE
JQuery
JQuery-Tests
Kernel
Kernel-Tests

Commit packageRename package Remove package

Instance Class Comment

1 renderButtonsOn: html
2 saveButton := html button.
3 saveButton
4 with: 'Save';
5 onClick: [self compile].
6 methodButtons := html span.

Save Remove method Method protocol References

Dolt PrintIt! InspectIt!

The screenshot shows the Amber Smalltalk web-based IDE. At the top, there's a toolbar with various icons and tabs. Below it is a navigation bar with links like 'Most Visited', 'Getting Started', 'Latest Headlines', etc. The main content area features a large orange sphere logo with a small bird icon inside, followed by the text 'Amber' and 'Smalltalk brought to the web'. Below this is a transcript workspace showing a list of packages and a code editor with some Smalltalk code. The code editor has tabs for 'Instance', 'Class', and 'Comment', with 'Instance' selected. At the bottom, there are buttons for 'Save', 'Remove method', 'Method protocol', 'References', and three utility buttons: 'Dolt', 'PrintIt!', and 'InspectIt!'.

<http://amber-lang.net/index.html>

mozilla

Some ES.next Features

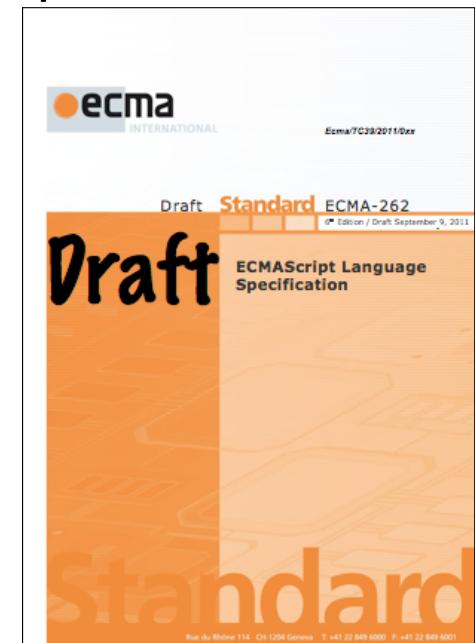
- Block scoping: let, const, functions
- Default value parameters, rest parameters, spread operator

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

- Destructuring in assignment, declarations and formal parameters.

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

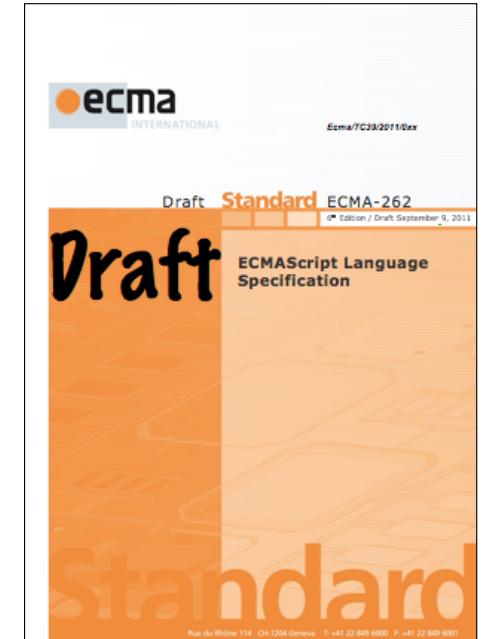
<http://wiki.ecmascript.org/doku.php?id=harmony:proposals>



mozilla

Major ES.next Enhancements

- Modules
- Sanding-boxing module loaders
- Proxy Objects – low level behavioral intercession
- Control abstraction via iterators and generators
- Array comprehensions, String interpolation
- super references
- Encapsulated state via gensym-like private names
- Better support for class-style inheritance



mozilla

New Object Composition Operators

<| “subclass” . { } “extend”

```
let test1 = {first: 1} <| {second: 2}; console.log(test1.first); //1  
  
let test2 = test1 <| [0,2,3];                                         console.log(test2.first); //1  
test2[49] = "I'm a real array";                                         console.log(test2.length); //50  
  
let test3 = test2 <| function() {};                                 console.log(test3.first); //1  
  
test3.{third: 3,  
      sum() {return this.first+this.second+this.third}  
    };  
  
console.log(test3.third) //3  
Console.log(test3.sum()) //6
```

mozilla

A “Class” pattern

```
const className = superClass <| function(/*constructor parameters */){  
    //constructor body  
    super.constructor(/*arguments to super constructor */);  
    this.{  
        //per instance property definitions  
    }  
}.prototype.{  
    //shared instance properties defined on prototype  
}.constructor.{  
    //class (ie, constructor) properties  
};
```

mozilla

Pattern-based “Classes”?

```
const ArrayedCollection = SequenceableCollection <| function(elements=0) {  
    super.constructor(elements);  
}.prototype.{  
    get size() {return this.basicSize}  
    at(index) {return this.basicAt(Math.floor(index))}  
    atPut(index,value) {return this.basicAtPut(Math.floor(index),value)}  
    add(newObject) {this.shouldNotImplement()}  
}.constructor.{  
    newWithAll(size,value) {  
        return (new this(size)).atAllPut(value);  
    }  
    with(...args) {  
        const newCollection = new this(args.length);  
        let i = 1;  
        args.forEach(function(element) {newCollection.atPut(i++,element)});  
        return newCollection;  
    }  
};
```

<https://github.com/allenwb/ESnext-experiments>

mozilla

Or Syntactic Classes?

```
class ArrayedCollection extends SequenceableCollection {  
    constructor (elements=0) {  
        super (elements);  
    }  
    get size() {return this.basicSize}  
    at(index) {return this.basicAt(Math.floor(index))}  
    atPut(index,value) {return this.basicAtPut(Math.floor(index),value)}  
    add(newObject) {this.shouldNotImplement()}  
    static newWithAll(size,value) {  
        return (new this(size)).atAllPut(value);  
    }  
    static with(...args) {  
        const newCollection = new this(args.length);  
        let i = 1;  
        args.forEach(function(element) {newCollection.atPut(i++,element)});  
        return newCollection;  
    }  
};
```

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

mozilla

What happening on the server?

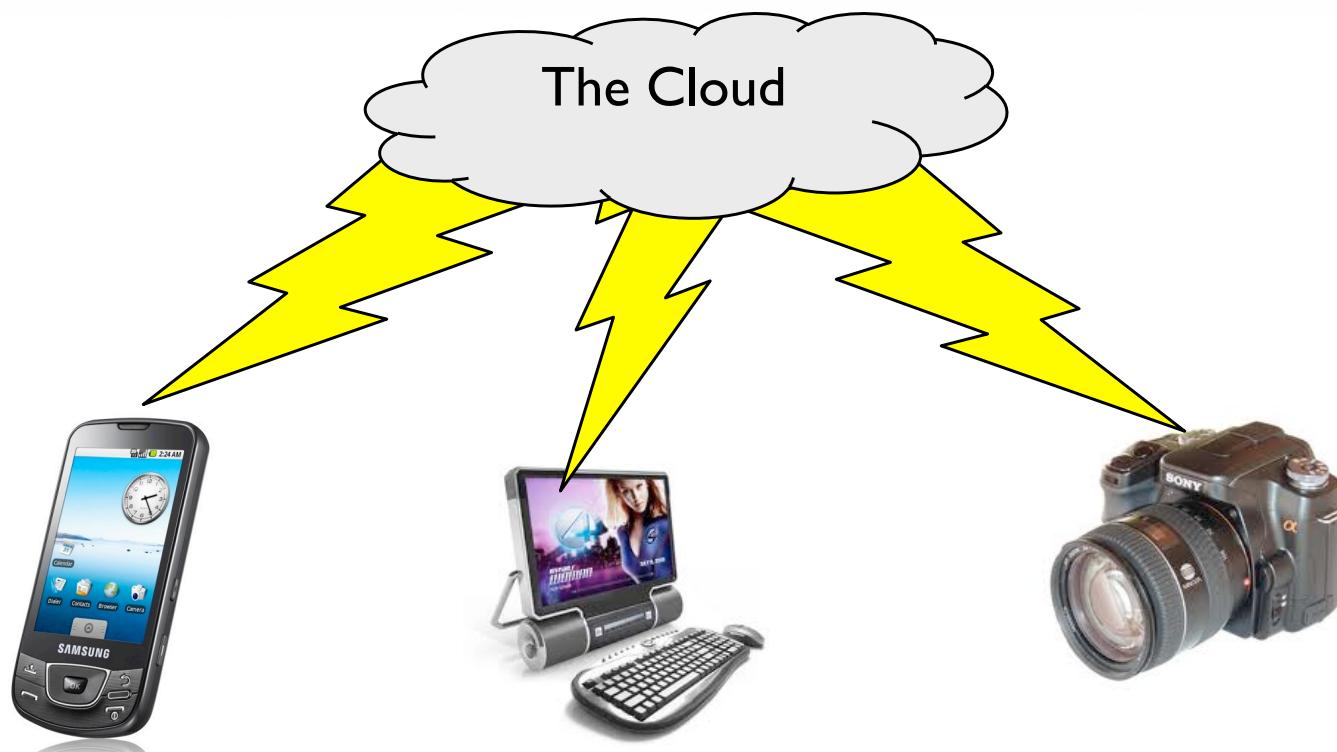
- The server-side of applications don't have the the “single platform” imperative of clients: Java, .Net, Python, Ruby
- Growing interest in also using JavaScript for the server portion of applications

Developers want to use same skills and technical resources on both client and server
- node.js (<http://nodejs.org/>) has exploded on the scene

```
//hello world server
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");
console.log('Server running at http://127.0.0.1:1337/');
```

mozilla

What About “The Cloud”



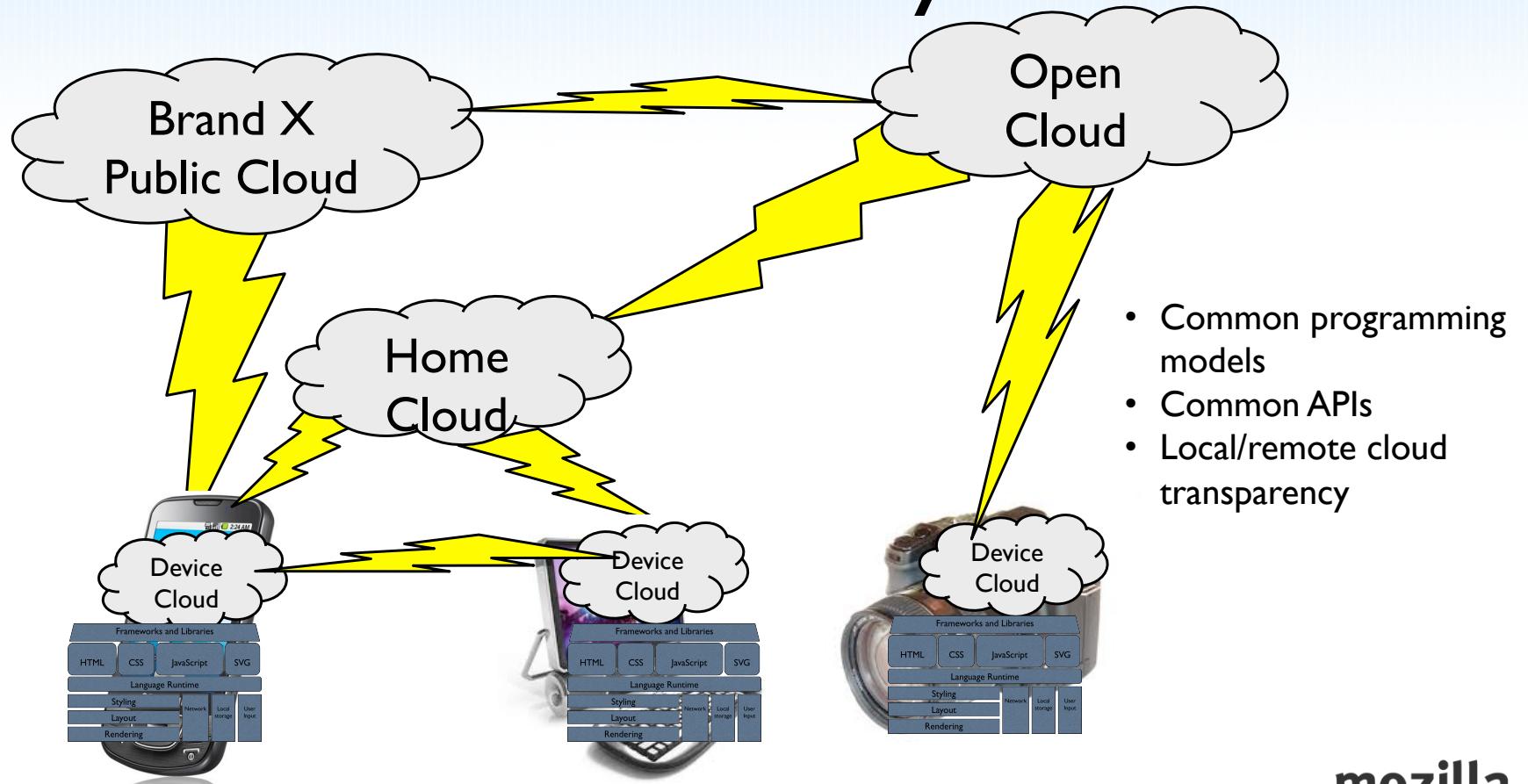
mozilla

Concurrency/Using Multicores

- The Browser App Platform uses a very simple no shared state concurrency model.
 - Web Workers
 - PostMessage and JSON data records
 - Event loops / Continuation passing style
- Essentially the same model as browser $\leftarrow\rightarrow$ webserver
- Can be fully distributed, not just client/server.

mozilla

Clouds All the Way Down?



mozilla

Users: Customers or Merchandise?

- Are we building devices and services that empower people
- Or, are we just trying to attract their attention so we can turn them into monetizable assets?

mozilla

User Sovereignty

- How can we keep the user in control?
- Some of the key issue,
 - Identity
 - Privacy
 - Freedom to use what we choose
 - Reliable storage of our digital assets
 - Durability of data and applications

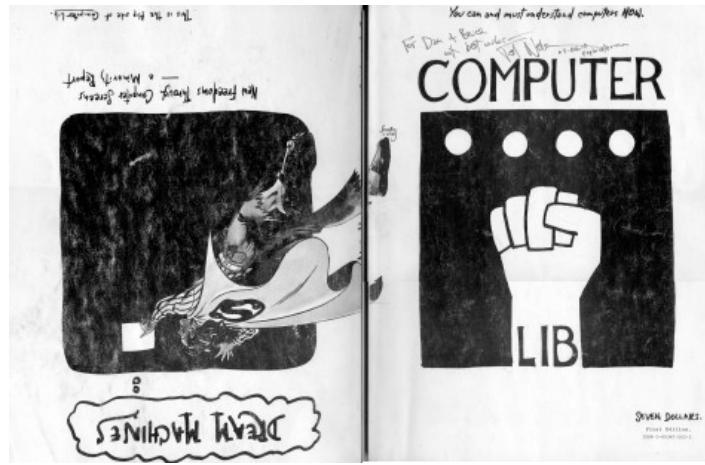
mozilla

We Need Vision, Not Strategy



mozilla

We Need Vision, Not Strategy



We Need Revolutionaries & Dreamers
mozilla

This is the most exciting time
for the computing industry
since the 1980's.



Enjoy it!

mozilla

Follow up

<http://www.wirfs-brock.com/allen/posts/category/post-pc-computing>

mozilla