A Brief History of JavaScript

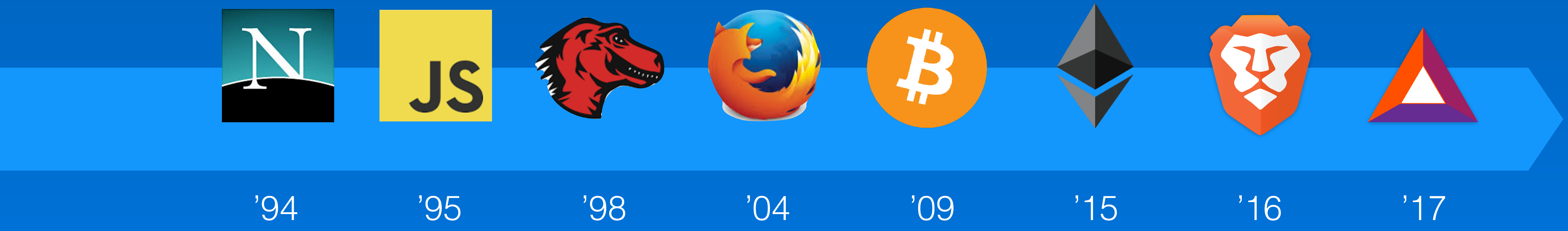
Brendan Eich
Brave Software
[@BrendanEich](https://twitter.com/BrendanEich)

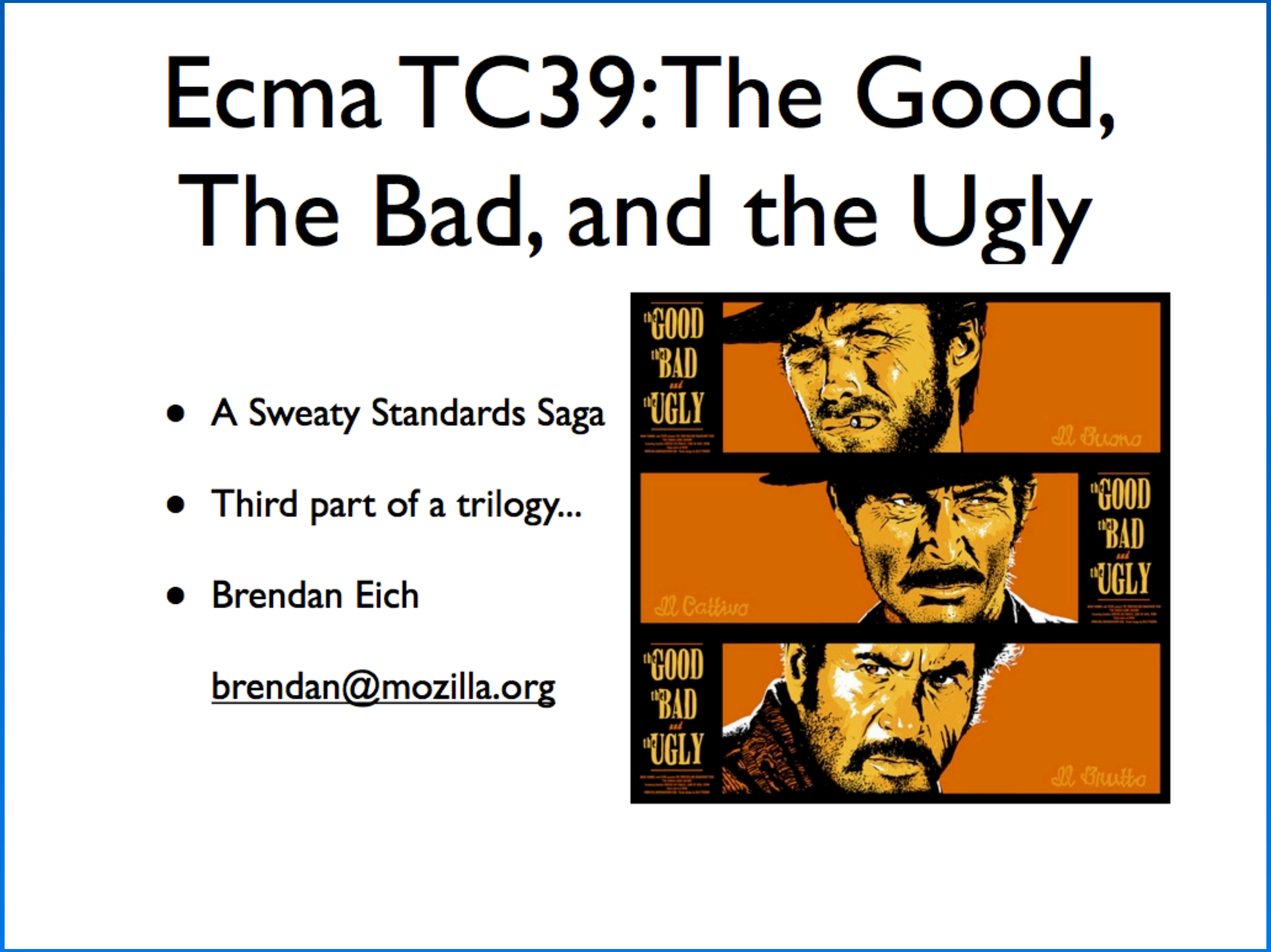
brave

Standardization

"Things that are impossible just take longer."
- Ian Hickson (HTML5 Editor)

Things have changed a little since I created JavaScript in ten days in May 1995





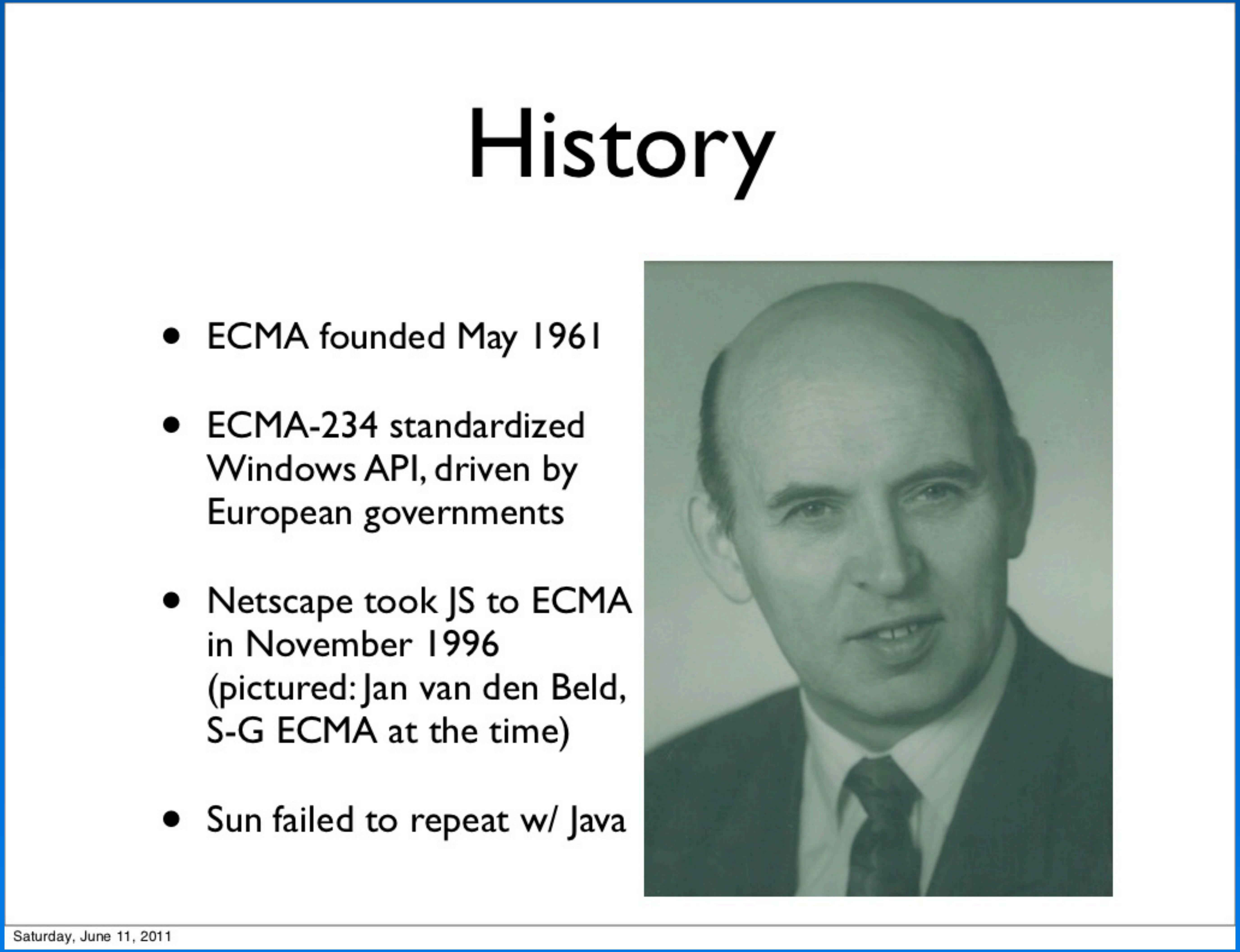
Ecma TC39: The Good, The Bad, and the Ugly

- A Sweaty Standards Saga
- Third part of a trilogy...
- Brendan Eich

  brendan@mozilla.org

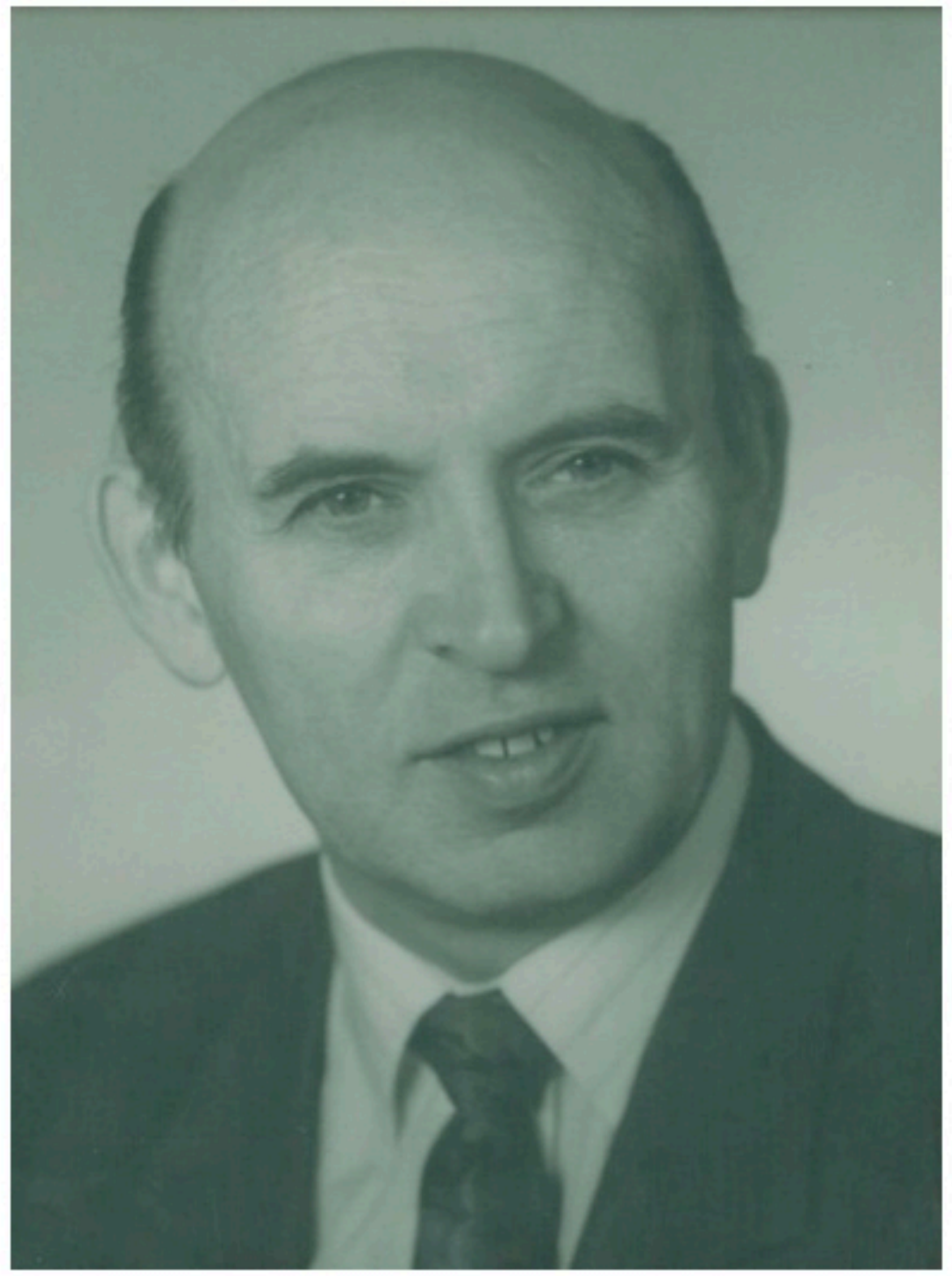
Standards Committees

Saturday, June 11, 2011

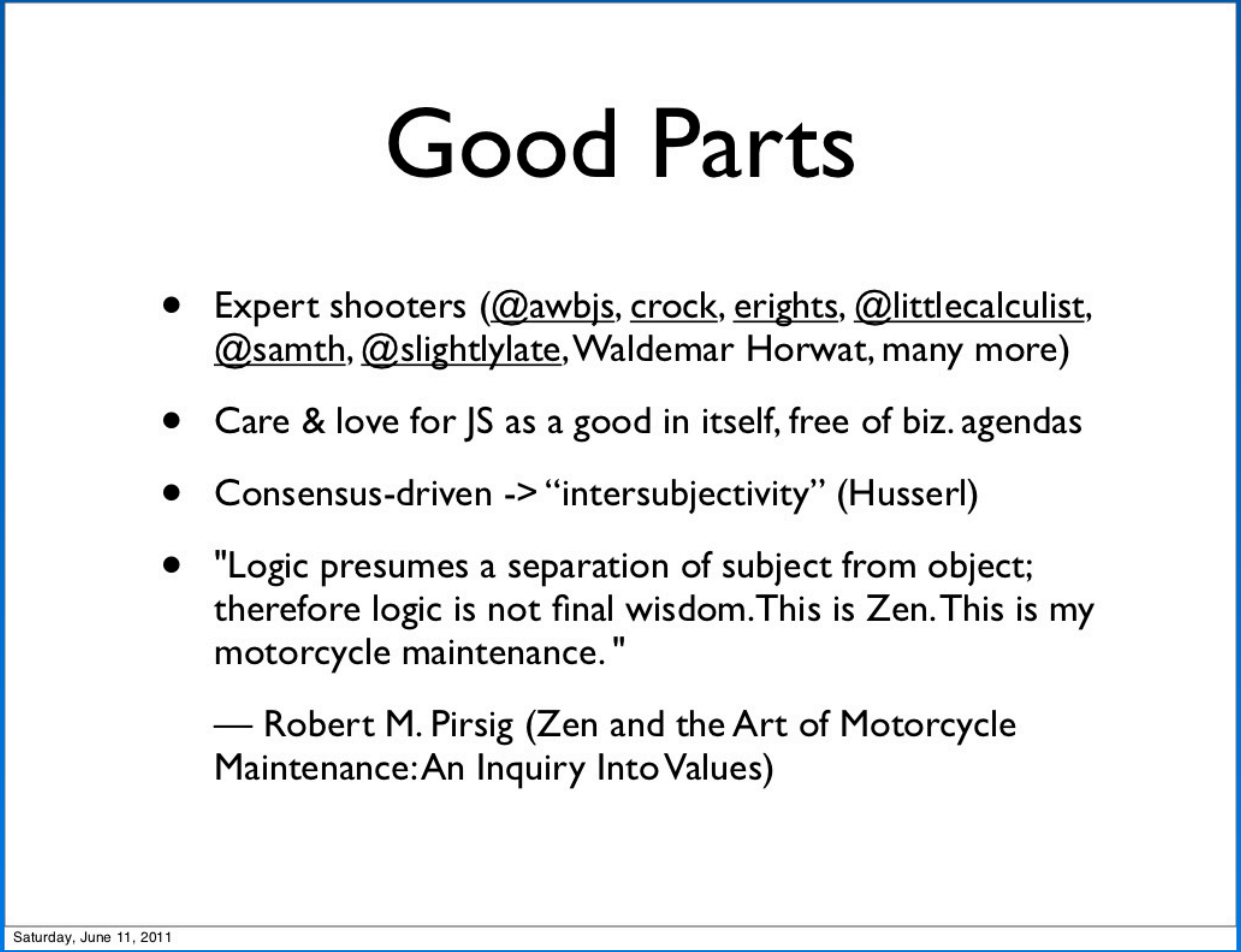


History

- ECMA founded May 1961
- ECMA-234 standardized Windows API, driven by European governments
- Netscape took JS to ECMA in November 1996 (pictured: Jan van den Beld, S-G ECMA at the time)
- Sun failed to repeat w/ Java



Saturday, June 11, 2011



Good Parts

- Expert shooters (@awbjs, crock, erights, @littlecalculist, @samth, @slightlylate, Waldemar Horwat, many more)
- Care & love for JS as a good in itself, free of biz. agendas
- Consensus-driven -> "intersubjectivity" (Husserl)
- "Logic presumes a separation of subject from object; therefore logic is not final wisdom. This is Zen. This is my motorcycle maintenance."

  — Robert M. Pirsig (Zen and the Art of Motorcycle Maintenance: An Inquiry Into Values)

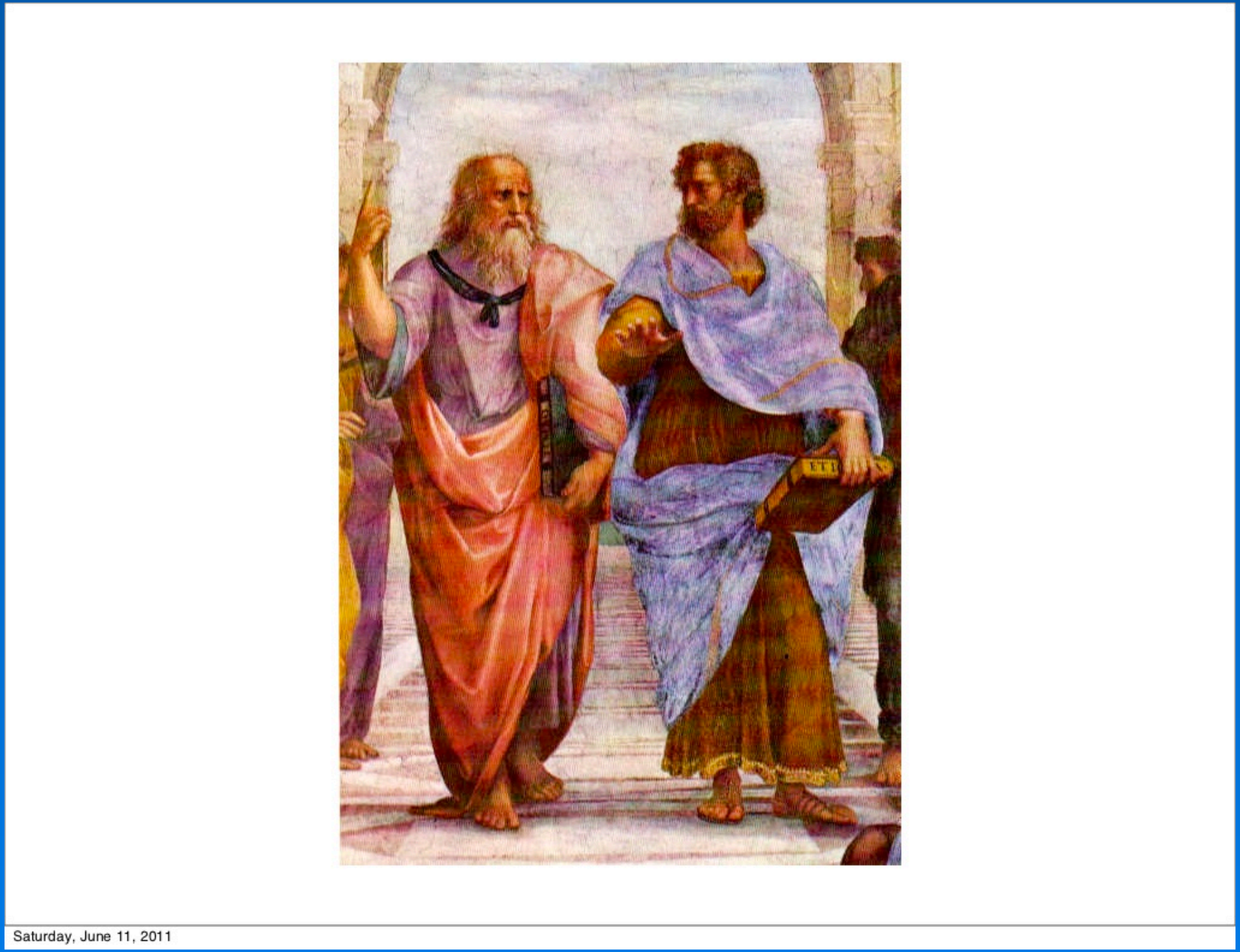
Saturday, June 11, 2011

The School of Athens

Saturday, June 11, 2011

FOR A BETTER WEB

Crock & Me

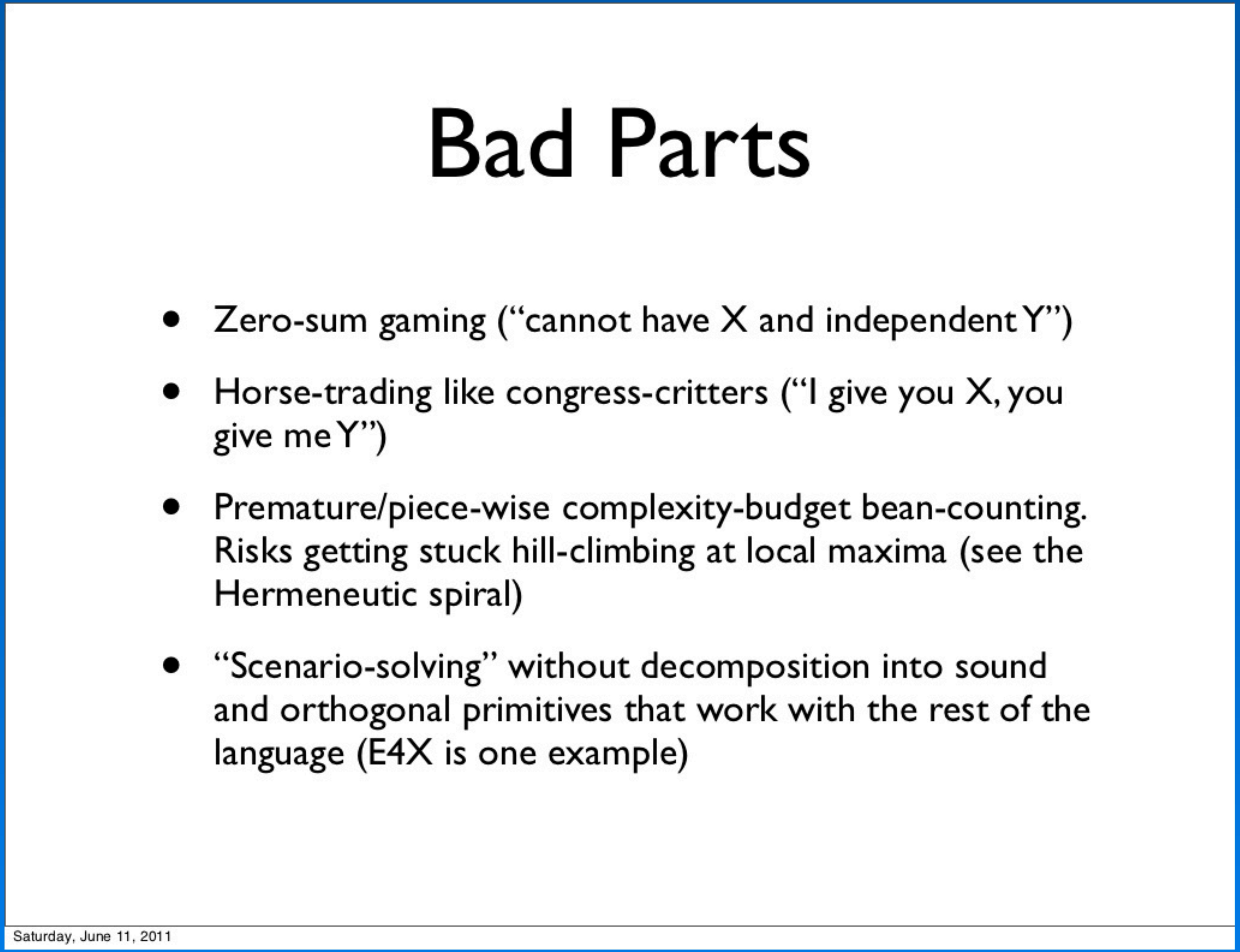


Saturday, June 11, 2011

FOR A BETTER WEB

The Bad

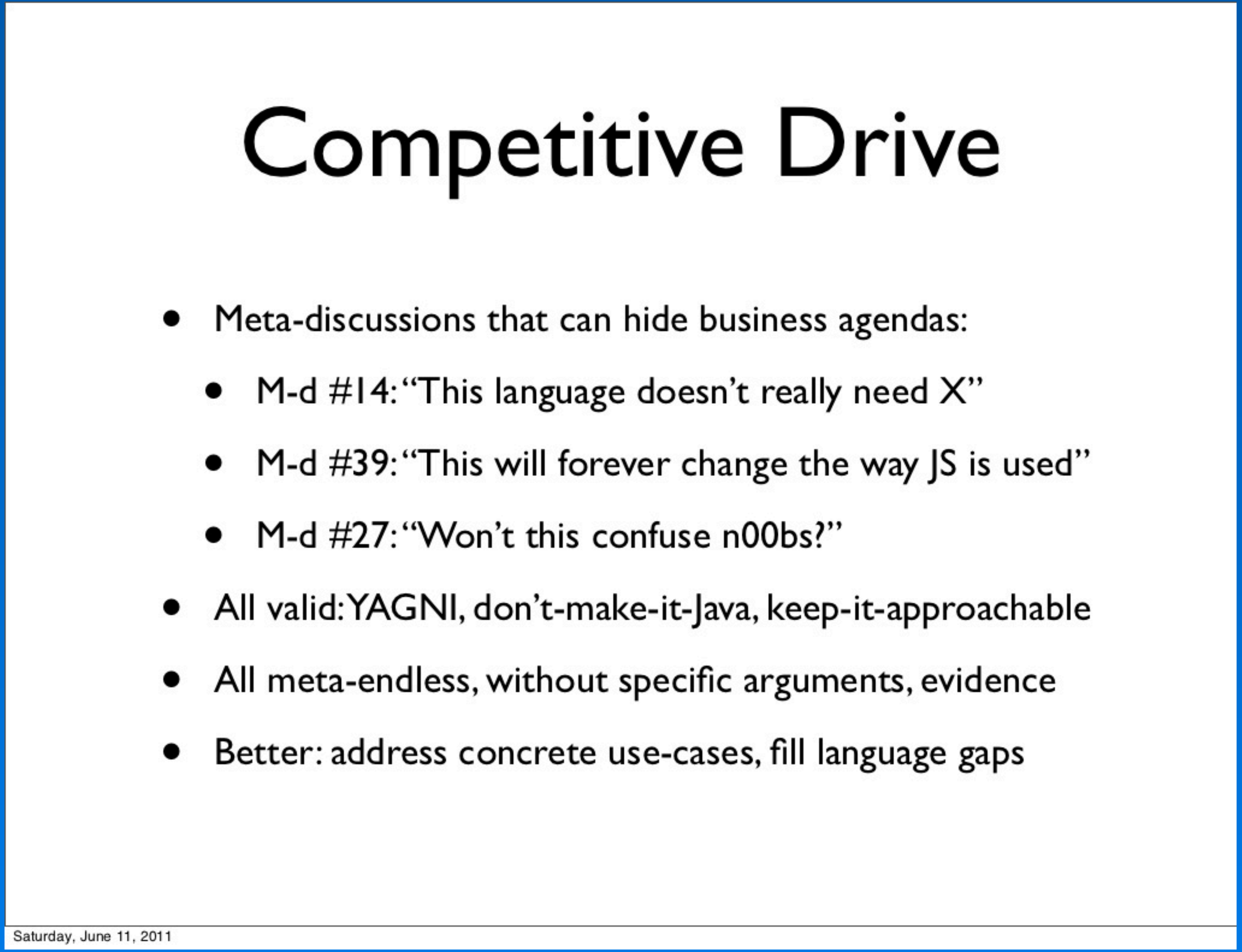
Saturday, June 11, 2011



Bad Parts

- Zero-sum gaming ("cannot have X and independent Y")
- Horse-trading like congress-critters ("I give you X, you give me Y")
- Premature/piece-wise complexity-budget bean-counting. Risks getting stuck hill-climbing at local maxima (see the Hermeneutic spiral)
- "Scenario-solving" without decomposition into sound and orthogonal primitives that work with the rest of the language (E4X is one example)

The Ugly



Competitive Drive

- Meta-discussions that can hide business agendas:
  - M-d #14: "This language doesn't really need X"
  - M-d #39: "This will forever change the way JS is used"
  - M-d #27: "Won't this confuse n00bs?"
- All valid: YAGNI, don't-make-it-Java, keep-it-approachable
- All meta-endless, without specific arguments, evidence
- Better: address concrete use-cases, fill language gaps

Saturday, June 11, 2011

ECMA-262 Editions

- 1997: ES1, based on JS1 — no closures, weak arrays
- 1998: ES2, just the ISO version of ES1
- 1999: ES3, based on JS1.2 — closures, arrays, do-while, switch, try-catch, regular expressions, Unicode (UCS-2)
- 2008: ES4, mothballed; many proposals made it to ES6
- 2009: ES5, formerly ES3.1, "no new *de jure* syntax", getters/setters, Object.defineProperty etc.
- 2015: ES6/2015, much of ES4 but no types, e.g. class, iterators and for-of, modules

The Back-Story

- 1995: JS1 — “come do Scheme in the browser!” j/k lol
- 1996-7: JS1.2 — closures, arrays, do-while, switch, try-catch, regular expressions from Perl 4, strict == !=
- 2004: Firefox 1.0 restarted the browser market
- 2005: I restarted Ecma TC39; with Macromedia allies we planned ES4 and did E4X (which prefigured JSX)
- 2008: V8, SpiderMonkey, JavaScriptCore — JS JITs
- 2010: Dash (now Dart) memo leaks at November TC39 meeting; not noticed until next spring

An asm.js example

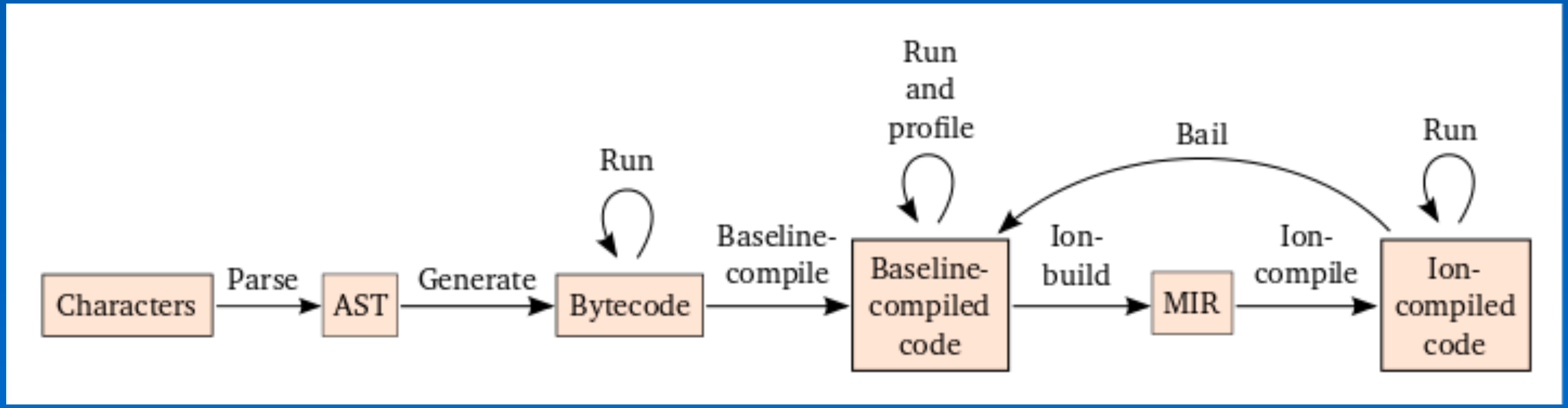
- C code:

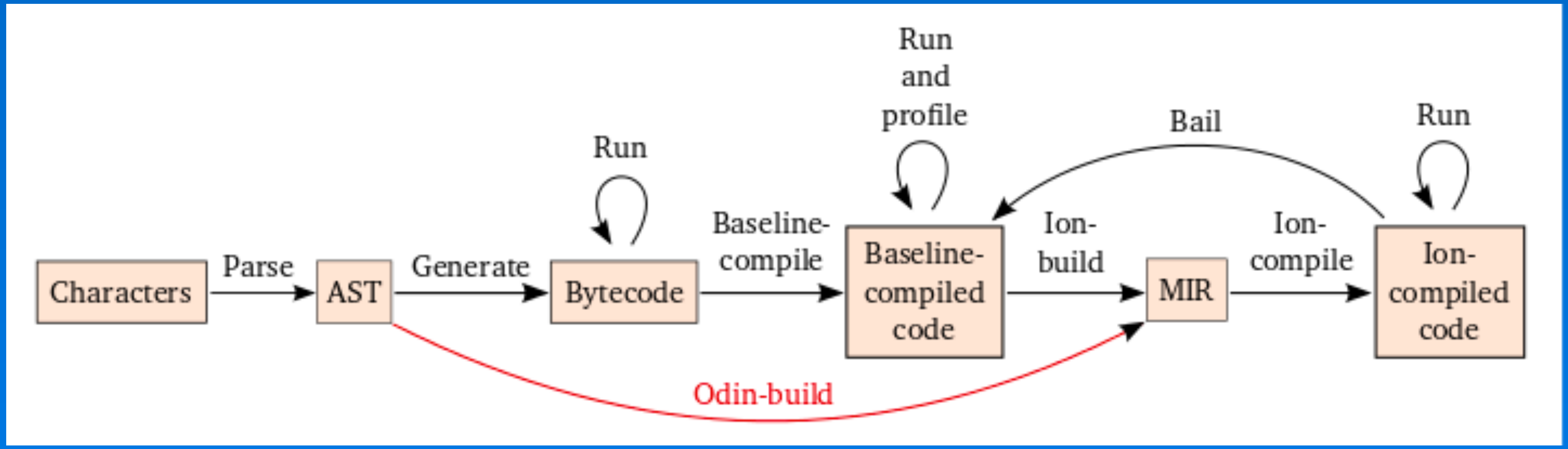
```
int f(int i) {          // i: 32-bit integer
  return i + 1;
}
```

- Generated `asm.js`:

```
"use asm";
function f(i) {         // i: any type
  i = i|0;              // coerce i to int32
  return (i + 1)|0;     // coerce return value
}
```

JIT vs. Ahead Of Time asm.js compilation

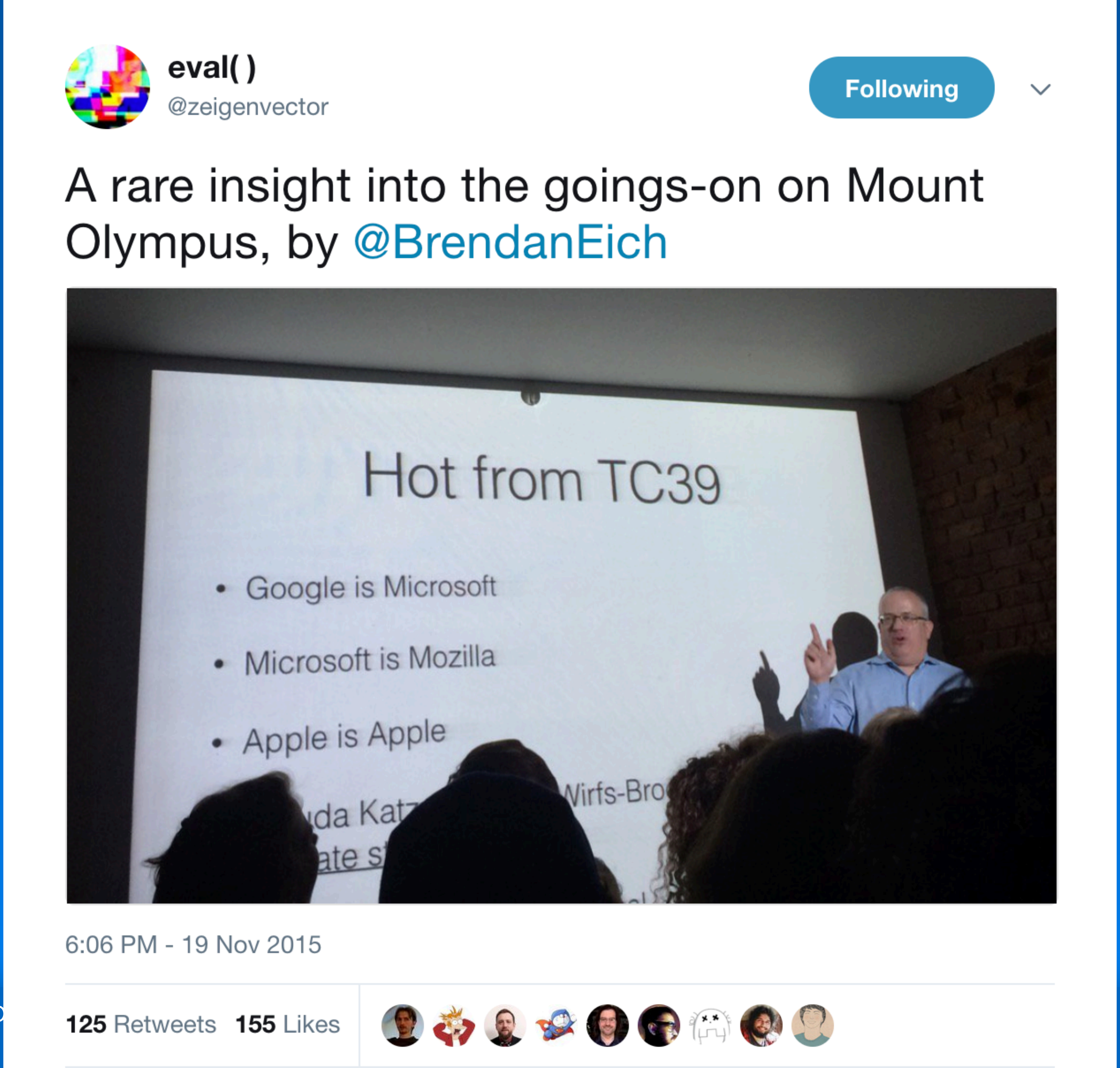




https://blog.mozilla.org/luke/2014/01/14/asm-js-aot-compilation-and-startup-performance/

More Back-Story

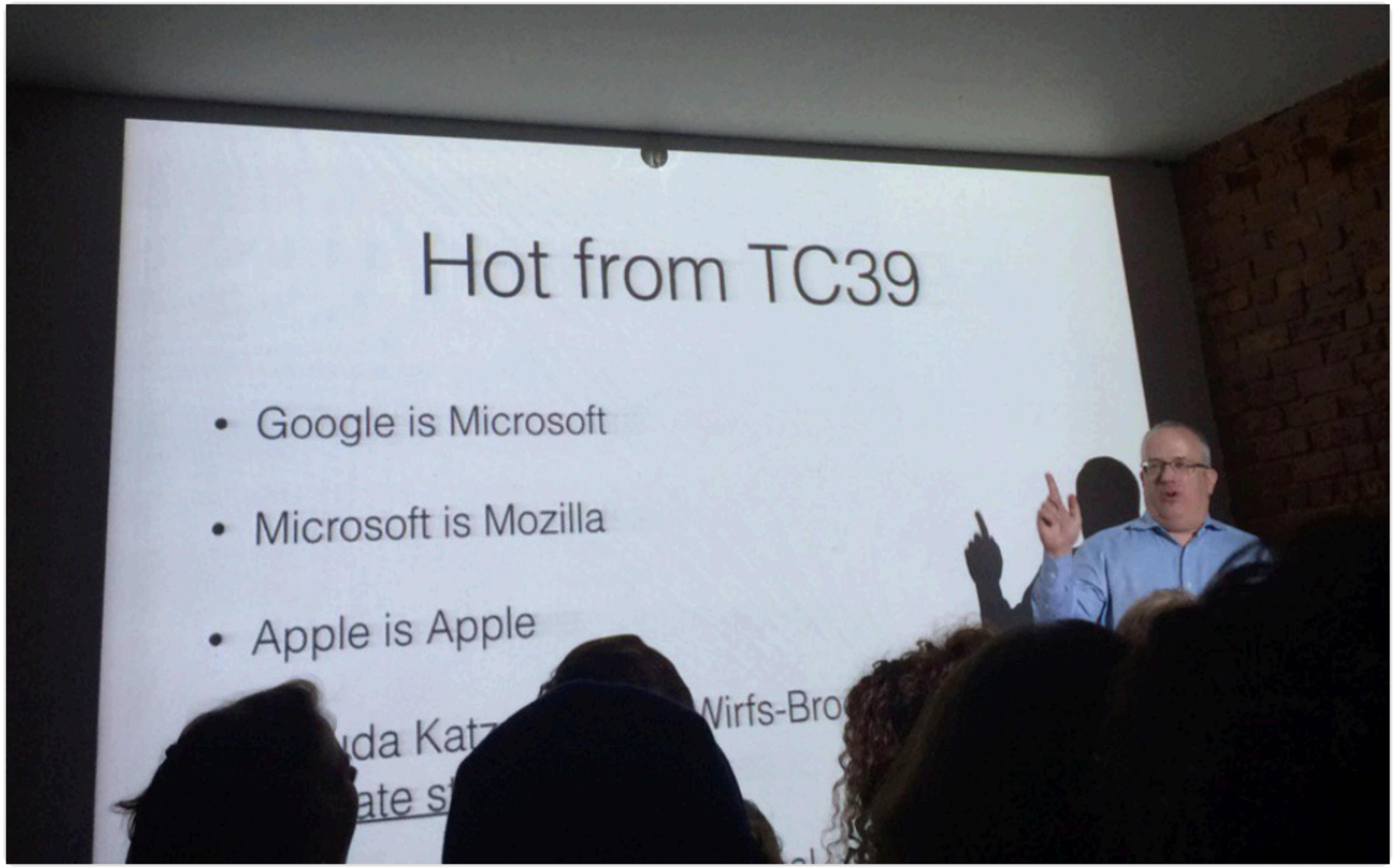
- December 2010: I recruit Allen Wirfs-Brock (ECMA-262 Editor) from Microsoft to Mozilla
- 2012: asmjs.org type system formalized; Epic Unreal Engine cross-compiled C++ to JS at 60fps in Firefox
- 2014: Babel.js (successor to 6to5) gets devs using ES6 early, acclimates many people to "compile to JS"
- 2015: Ecma TC39 moves to annuals, ES6 => ES2015
- March 2015: Google admits Dart won't ever go in Chrome
- December 2015: Microsoft open-sources ChakraCore



eval( )
@zeigenvector

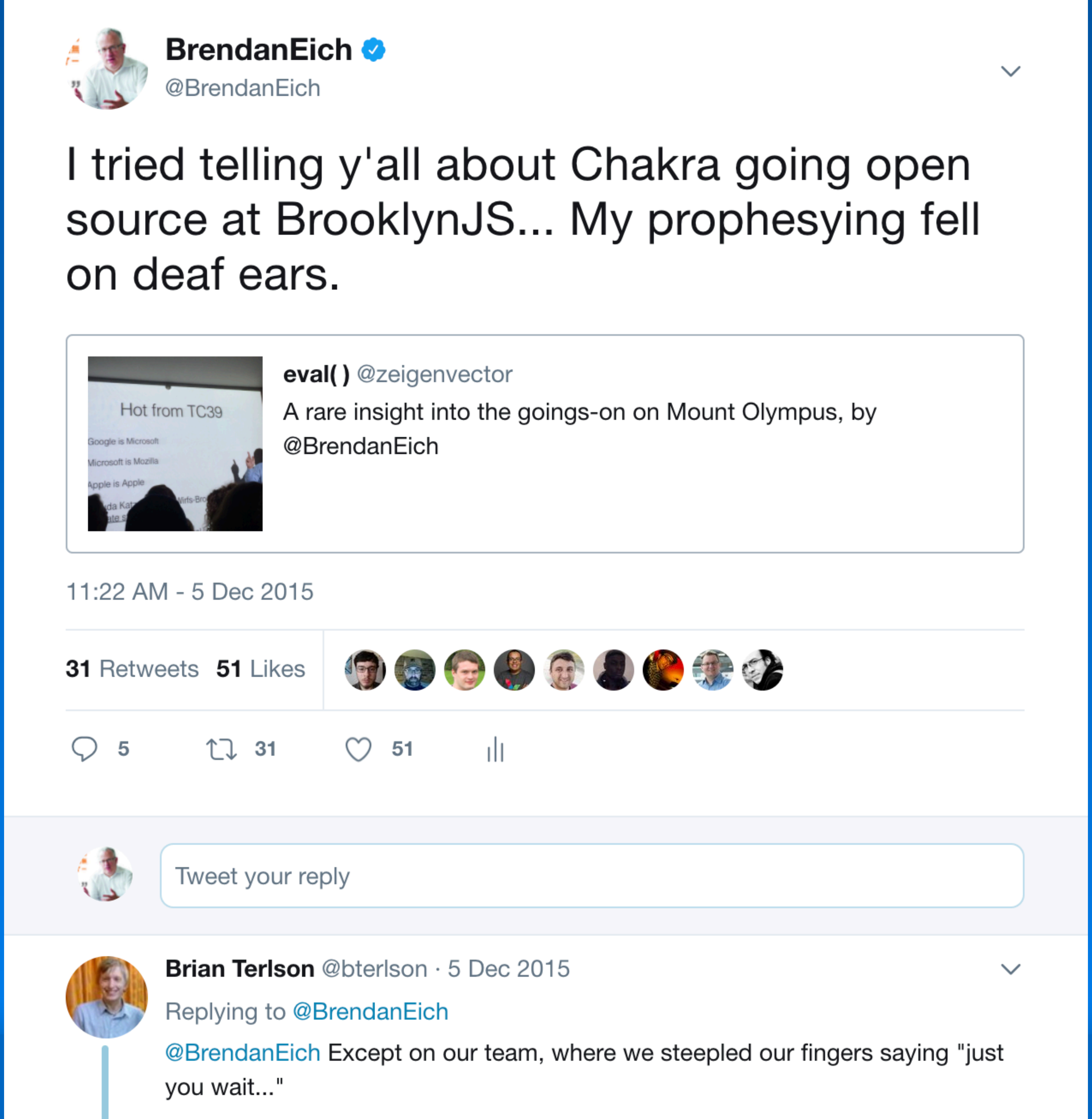
Following

A rare insight into the goings-on on Mount Olympus, by @BrendanEich



6:06 PM - 19 Nov 2015

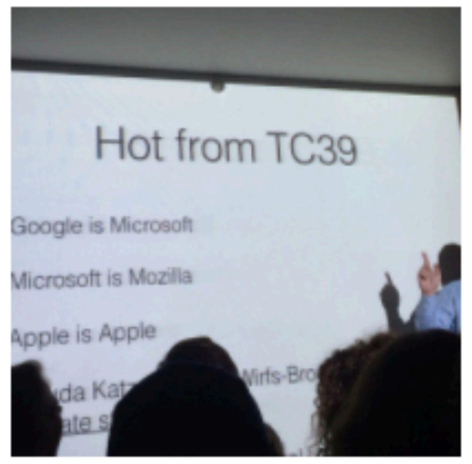
125 Retweets **155** Likes



BrendanEich @BrendanEich

I tried telling y'all about Chakra going open source at BrooklynJS... My prophesying fell on deaf ears.

eval( ) @zeigenvector
A rare insight into the goings-on on Mount Olympus, by @BrendanEich



11:22 AM - 5 Dec 2015

31 Retweets 51 Likes

5 31 51

Tweet your reply

Brian Terlson @bterlson · 5 Dec 2015
Replying to @BrendanEich
@BrendanEich Except on our team, where we steepled our fingers saying "just you wait..."

FOR

TC39: BigInt

- New value type to handle arbitrary precision integers
- Literal syntax: `4353923459876432589763587 9n`
- Operator overloading: `1n + 2n === 3n`
- Exceptions on mixed types: `1n + 2` throws `TypeError`
    - However, allow mixed comparisons using < and ==
- someObject[42n]: BigInt as distinct property key type
- BigInt.asUintN(N, b): wrap b between 0 and 2^{N-1}
- BigInt.asIntN(N, b): wrap b between -2^{N-1} and $2^{N-1}-1$

More BigInt

- JSON hooking via `BigInt.prototype.toJSON()`
- New typed arrays: `BigInt64Array`/`BigUint64Array`
- `DataView.prototype.getBigInt64/getBigUint64`
- Explainer: https://github.com/tc39/proposal-bigint
- Spec: https://tc39.github.io/proposal-bigint/
- Issues: https://github.com/tc39/proposal-bigint/issues

BigInt FTW

```
/*
 * Avoid 53-bit limit of JS’s default number
 * type. Thus fib(79) is 14472334024676221n,
 * not 14472334024676220.
 */
function fib(n) {
  let [a, b] = [0n, 1n];
  for (let i = 0; i < n; i++) {
    [a, b] = [b, a + b];
  }
  return a;
}
```

More ES Next

- Dynamic `import()` (spec)
- `Array.prototype.flatten/.flatMap` (spec)
- `let {x, y, ...z} = {x:1, y:2, a:3, b:4};` (spec)
- Private methods and accessors (spec)
- Asynchronous iteration: `for await of` (spec)
- RegExp lookbehind assertions (spec)
- RegExp Unicode property escapes (spec)
- RegExp named capture groups (spec)
- `/s` (`dotAll`) flag for regular expressions (spec)

Always bet on JS

- First they said JS couldn't be useful for building "rich Internet apps"
- Then they said it couldn't be fast
- Then they said it couldn't be fixed
- Then it couldn't do multicore/GPU
- Wrong every time!
- My advice: always bet on JS & WASM!

& Webpack lol for @TheLarkInn

Thank you