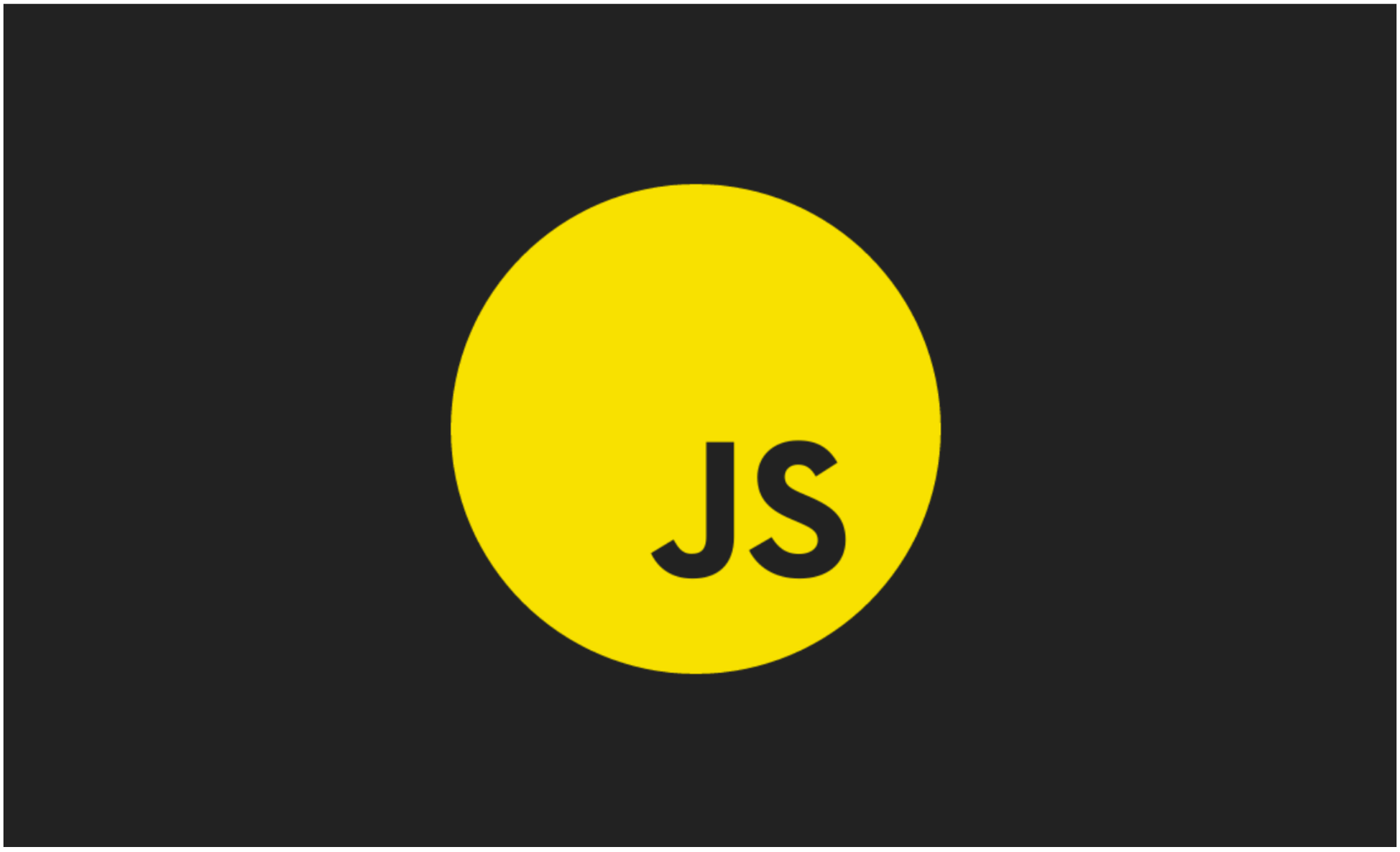
ECMAScript Harmony: Rise of the Compilers

Brendan Eich

Fluent 2015



Solar System of JS

Hats off to @shaunlebron — and to ClojureScript

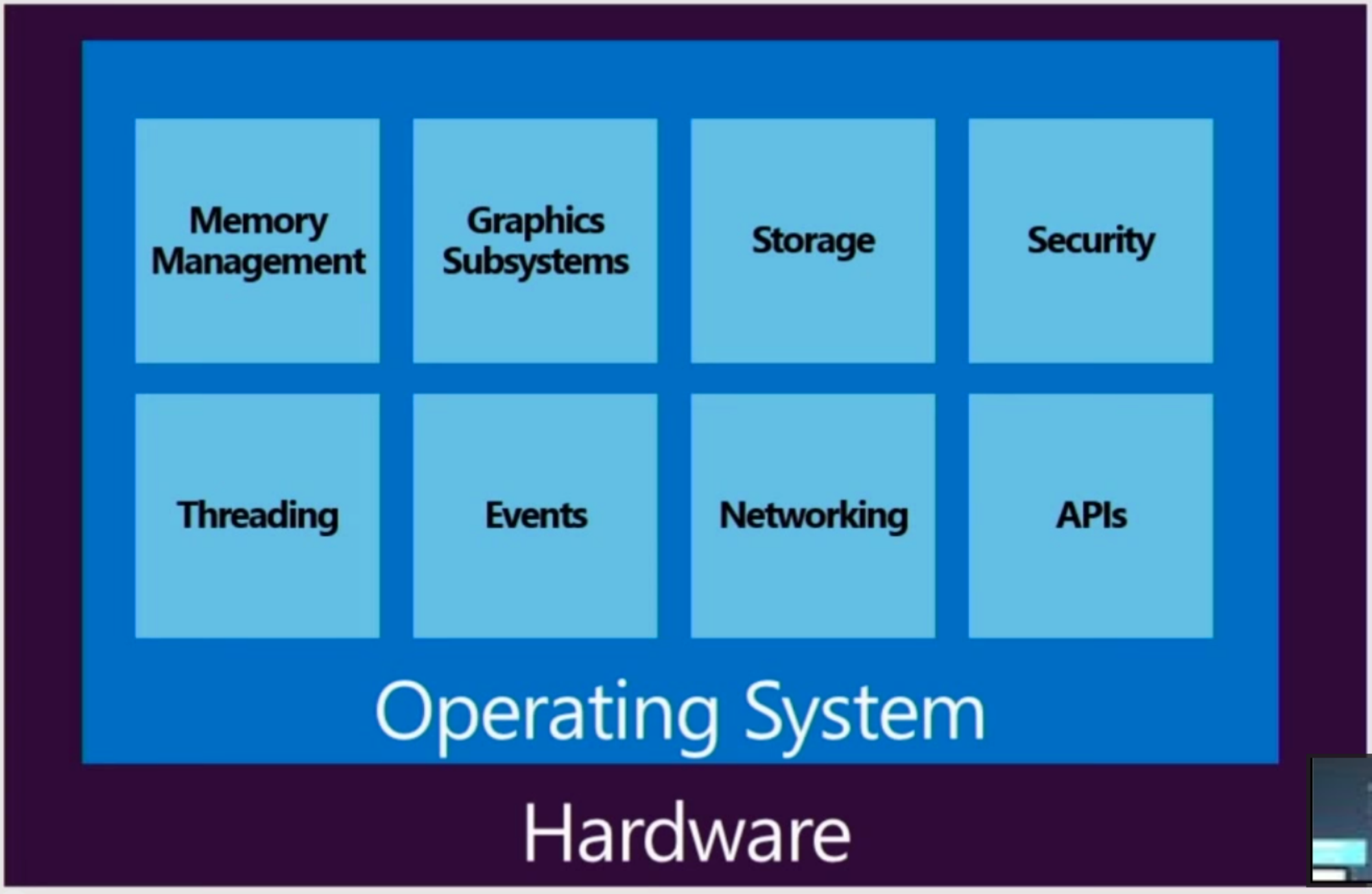
Paris

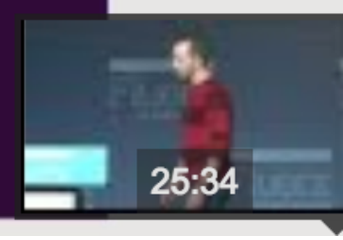
Where ES6 was approved by Ecma TC39
(but not in this building)

Inria Paris Roof Deck

Our view when we voted ES6 through TC39

What Can JS do?



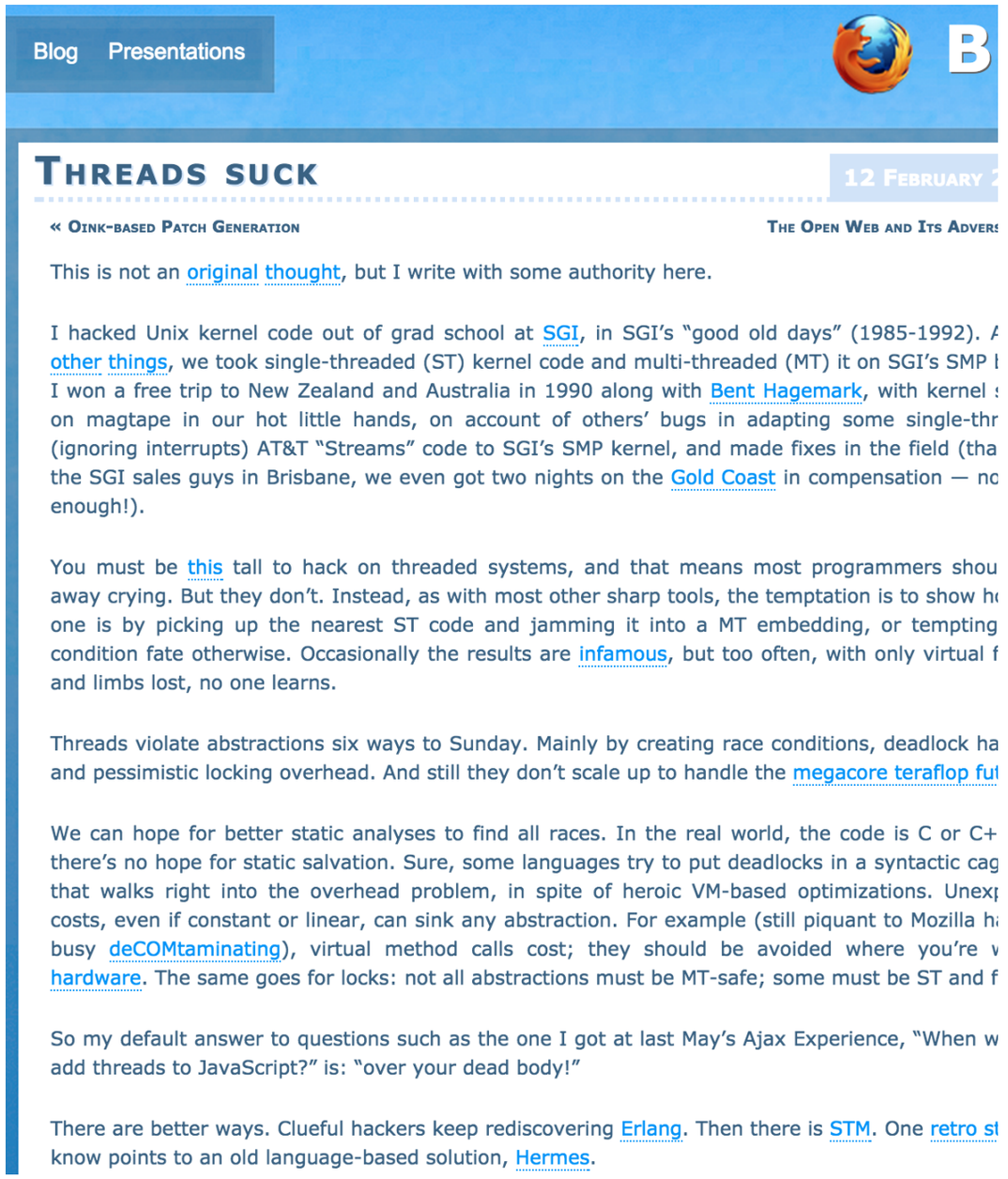


What JS Can't Do

- 64-bit integers (COMING IN ES7 <3<3<3)
- Safe stack allocation (e.g. as in Rust)
    - Compilers can use a typed array, but slowly
- Mixing objects and primitives in typed arrays
    - Proposed for ES7/2016 via typed objects
- Shared memory threads (as in C++)

Don’t Threads Suck?

- Yes — but not compiling C++ to JS sucks worse
- And C++ has threads
- Emscripten all the C++ things
- Other “Blub to JS” compilers require threads too
- No shared DOM or main thread memory



Blog Presentations

B

THREADS SUCK

12 FEBRUARY

« OINK-BASED PATCH GENERATION

THE OPEN WEB AND ITS ADVERS

This is not an original thought, but I write with some authority here.

I hacked Unix kernel code out of grad school at SGI, in SGI’s “good old days” (1985-1992). A other things, we took single-threaded (ST) kernel code and multi-threaded (MT) it on SGI’s SMP I I won a free trip to New Zealand and Australia in 1990 along with Bent Hagemark, with kernel s on magtape in our hot little hands, on account of others’ bugs in adapting some single-thr (ignoring interrupts) AT&T “Streams” code to SGI’s SMP kernel, and made fixes in the field (tha the SGI sales guys in Brisbane, we even got two nights on the Gold Coast in compensation — nc enough!).

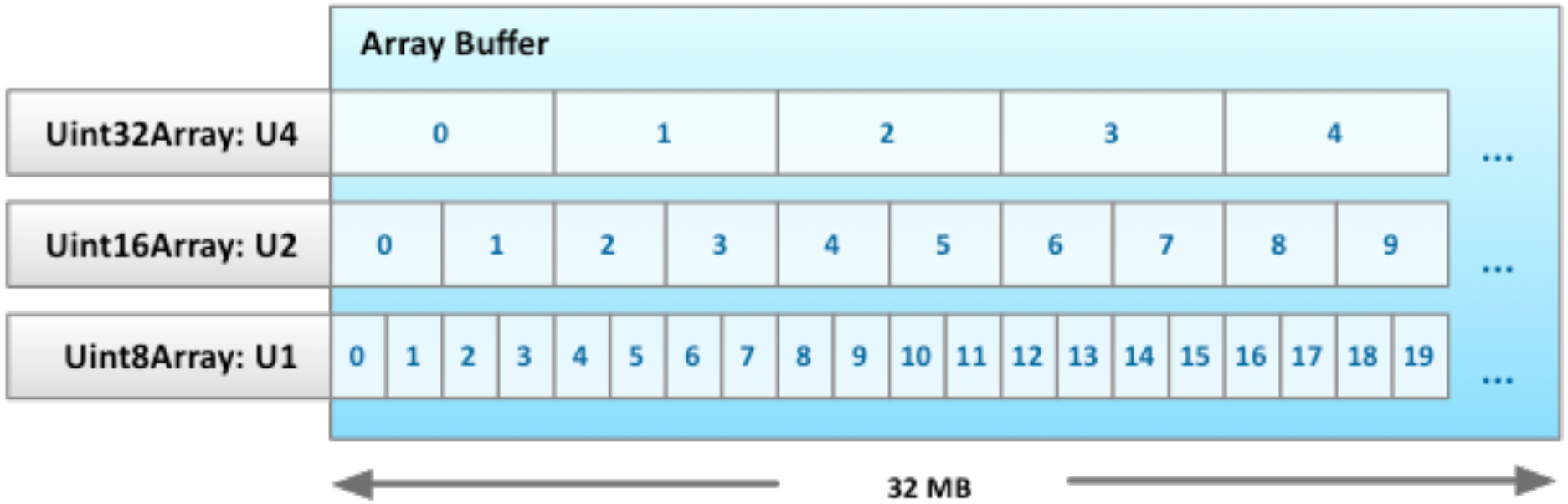
You must be this tall to hack on threaded systems, and that means most programmers shou away crying. But they don’t. Instead, as with most other sharp tools, the temptation is to show h one is by picking up the nearest ST code and jamming it into a MT embedding, or tempting condition fate otherwise. Occasionally the results are infamous, but too often, with only virtual f and limbs lost, no one learns.

Threads violate abstractions six ways to Sunday. Mainly by creating race conditions, deadlock ha and pessimistic locking overhead. And still they don’t scale up to handle the megacore teraflop fut

We can hope for better static analyses to find all races. In the real world, the code is C or C+ there’s no hope for static salvation. Sure, some languages try to put deadlocks in a syntactic cag that walks right into the overhead problem, in spite of heroic VM-based optimizations. Unexp costs, even if constant or linear, can sink any abstraction. For example (still piquant to Mozilla h busy deCOMtaminating), virtual method calls cost; they should be avoided where you’re v hardware. The same goes for locks: not all abstractions must be MT-safe; some must be ST and f

So my default answer to questions such as the one I got at last May’s Ajax Experience, “When w add threads to JavaScript?” is: “over your dead body!”

There are better ways. Clueful hackers keep rediscovering Erlang. Then there is STM. One retro st know points to an old language-based solution, Hermes.



Typed Arrays (ES6)

Originated in WebGL

Copy, or Hand Off

You can copy a typed array, and you can hand off its buffer memory across a Web Worker boundary.

But in HTML5, you cannot share memory among workers — UNTIL NOW!

SharedWorker

```
var worker = new SharedWorker(filename);

const sentMessage = "ping";
var receivedMessage;
var receivedError;
```

```
worker.port.onmessage = function (event) {
 receivedMessage = event.data;
};

worker.onerror = function (event) {
  receivedError = event.message;
};

worker.port.postMessage(sentMessage);
```

SharedArrayBuffer

```
var buffer = new SharedArrayBuffer(1<<20);

var bytes = new SharedUint8Array(buffer);
var words = new SharedUint32Array(buffer);

// etc. as with ES6 typed arrays, but Shared

sharedWorkers.forEach(worker =>
  worker.port.postMessage(“start”, [buffer])
);
```

(draft spec gdoc)

Atomics

```
Atomics.compareExchange(sta, index, oldvalue,
                        newvalue)
Atomics.load(sta, index)
Atomics.store(sta, index, value)
Atomics.add(ia, index, value)
Atomics.sub(ia, index, value)
Atomics.and(ia, index, value)
Atomics.or(ia, index, value)
Atomics.xor(ia, index, value)
Atomics.exchange(ia, index, value)
Atomics.isLockFree(size)
Atomics.futexWait(i32a, index, value, timeout)
Atomics.futexWake(i32a, index, count)
Atomics.futexWakeOrRequeue(i32a, index1, count,
                           value, index2)
```

In Firefox, + now Chrome

JF Bastien
@jfbastien

Following

Intent to implement: SharedArrayBuffer.
groups.google.com/a/chromium.org…
Cool work from @binjimint!

RETWEETS 25 FAVORITES 34

5:32 PM - 18 Apr 2015

Reply to @jfbastien @binjimint

Aras Pranckevičius @aras_p · Apr 19

Demos

(PWD, DT2)

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