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?

Memory Management  Graphics Subsystems  Storage  Security

Threading  Events  Networking  APIs

Operating System

Hardware
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
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

```javascript
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

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

5:32 PM - 18 Apr 2015
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