State of JS Implementations, 2014 Edition

webengineshackfest.org

Andy Wingo

Agenda

History

New things

A brief history of JS

1996-2008: slow

2014: fastish

A brief history of JS

1996-2008: slow

2014: fastish

Environmental forcing functions

Visiting a page == installing an app

Cruel latency requirements

JS: speed via dynamic proof

“Adaptive optimization”

A revival of compilation techniques pioneered by Smalltalk, Self, Strongtalk, Java

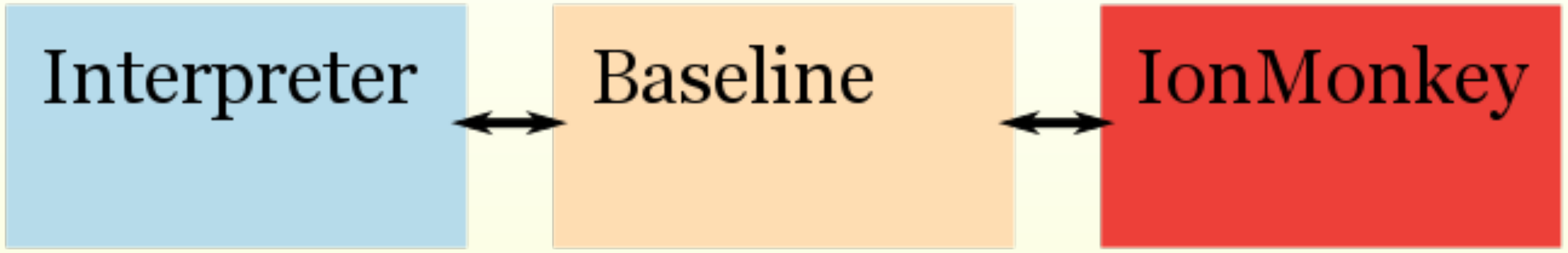
`expr ifTrue: block`

Inlining key for performance: build sizable proof term

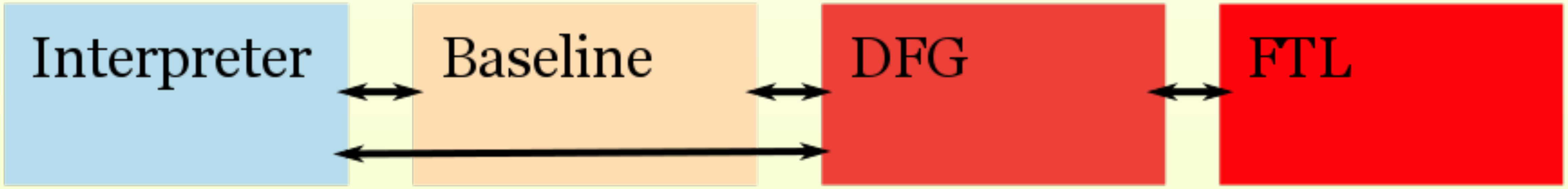
JS focus: *low-latency* adaptive optimization (fast start)

❧ lazy parsing and compilation

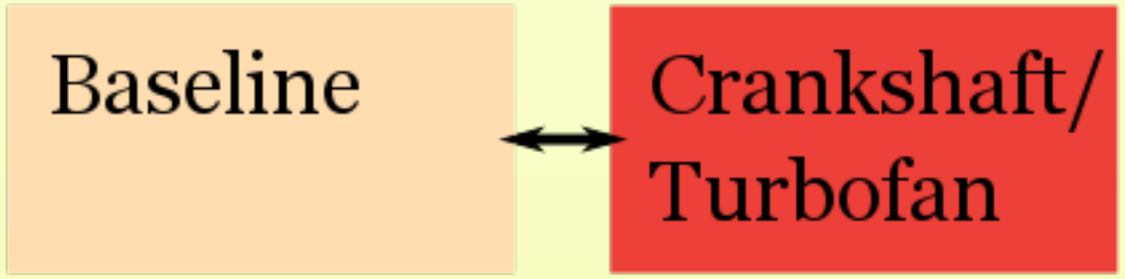
SpiderMonkey (Firefox)



JavaScriptCore (WebKit, Safari)



V8 (Chrome)



All about the tiers

"Method JIT compilers"; Java's HotSpot is canonical comparison

The function is the unit of optimization

`asm.js` code can start in IonMonkey / Turbofan; embedded static proof pipeline

Optimizing compiler awash in information

Operand and result types

Free variable values

Global variable values

Sets of values: mono-, poly-, mega-morphic

Optimizations: An inventory

Inlining

Code motion: CSE, DCE, hoisting, sea-of-nodes

Specialization

- ❧ Numeric: int32, uint32, float, ...
- ❧ Object: Indexed slot access
- ❧ String: Cons, packed, pinned, ...

Allocation optimization: coalescing, scalar replacement, sinking

Register allocation

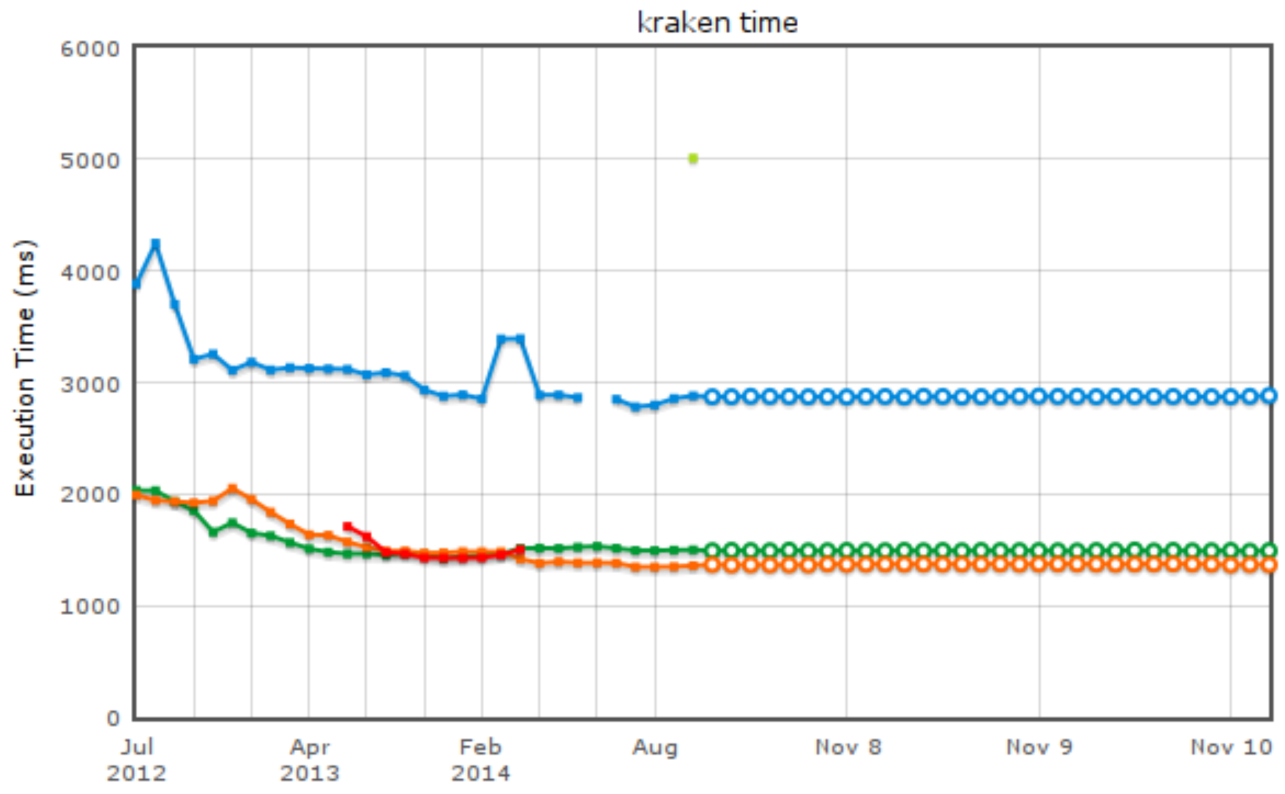
Dynamic proof, dynamic bailout

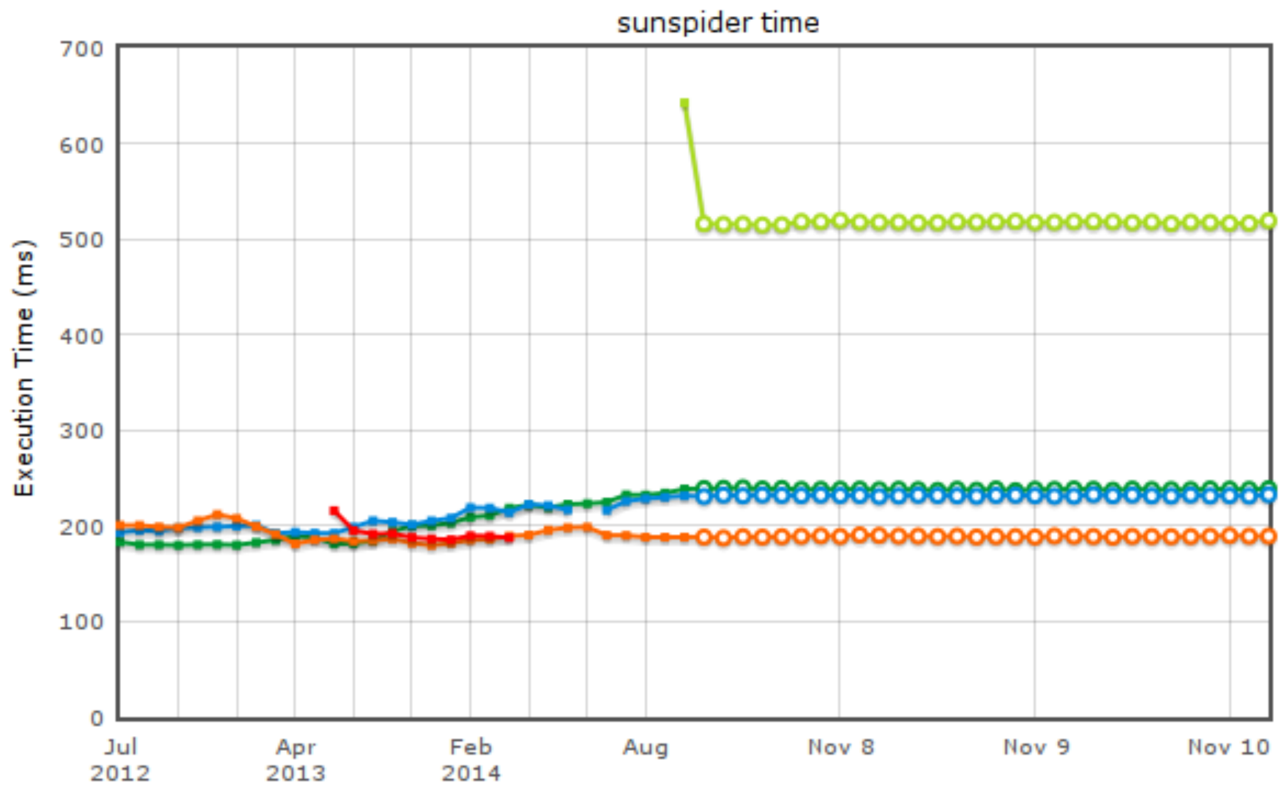
Compilation is proof-driven term specialization

Dynamic assertions: the future will be like the past

Dynamic assertion failure causes proof invalidation: abort (“bailout”) to baseline tier

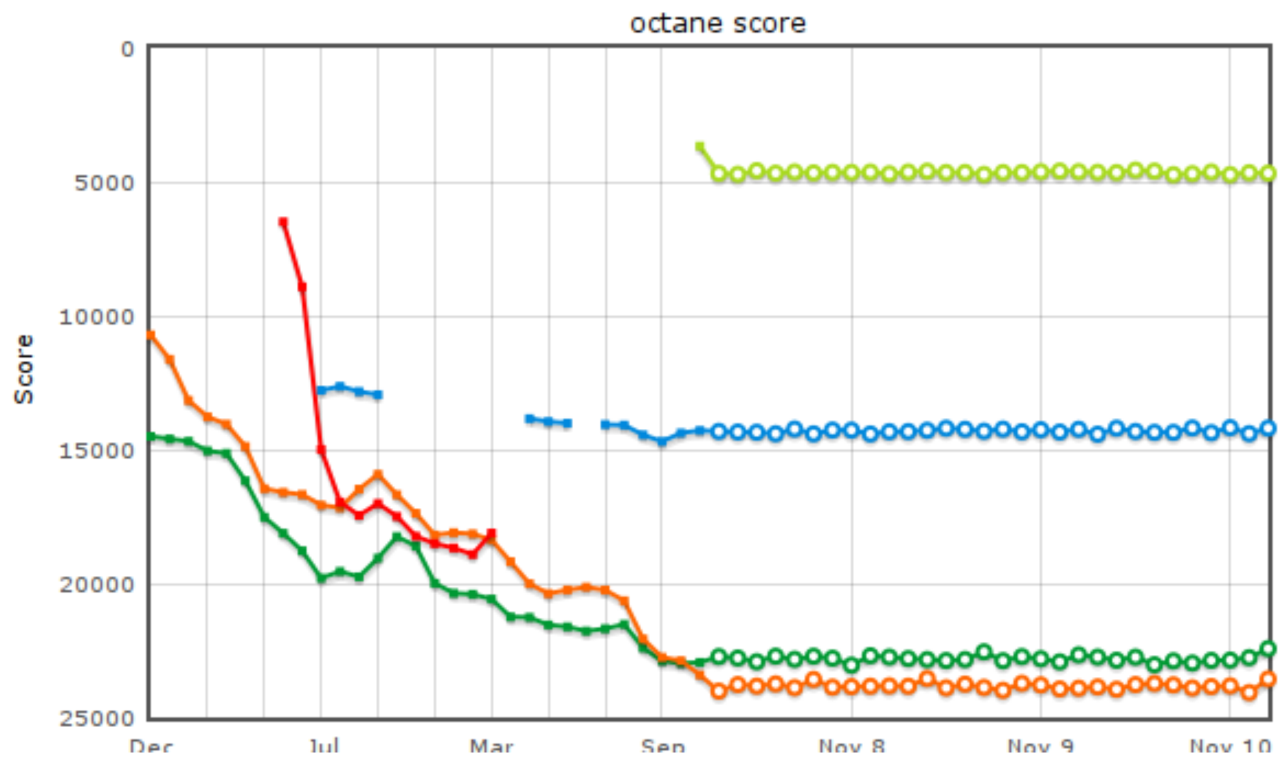
Bailout enables static compilation techniques (FTL)





🟩 Chrome (v8)
🟦 Safari (jsc)
🟧 Firefox (Ion)
🟥 Firefox (Ion, GGC old)
🟩 Chrome (v8-turbofan)

Machines »
Breakdown »
About »



A brief history of JS

1996-2008: slow

2014: fastish

...via adaptive optimization.

New things in 2014

New things in 2014

SM, JSC, V8

First perf, then features

SpiderMonkey perf

SM won Octane!

Landing of precise GC, then generational GC

`https://blog.mozilla.org/javascript/2013/07/18/clawing-our-way-back-to-precision/`

Compacting GC in the works

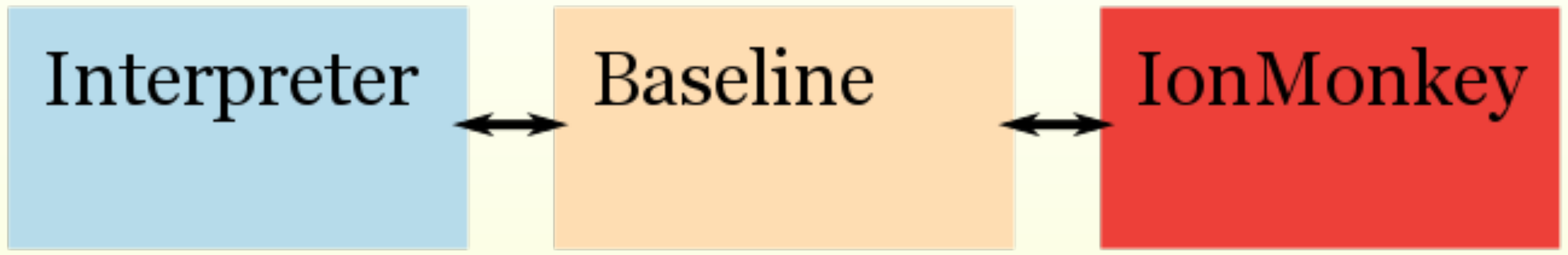
Lots of Ion work

JSC perf

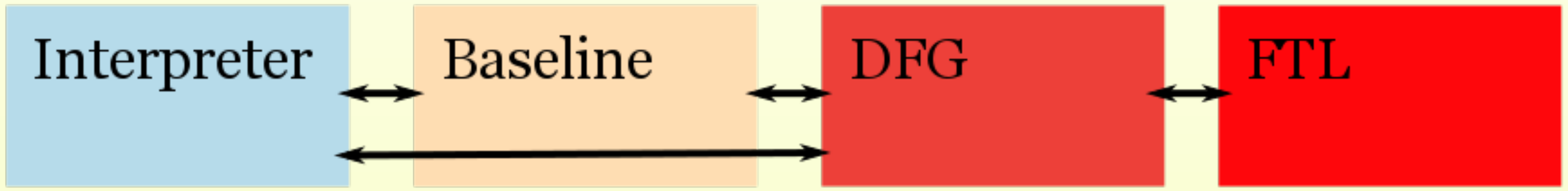
(I am less knowledgeable here)

“Fourth-tier LLVM” (FTL) JIT

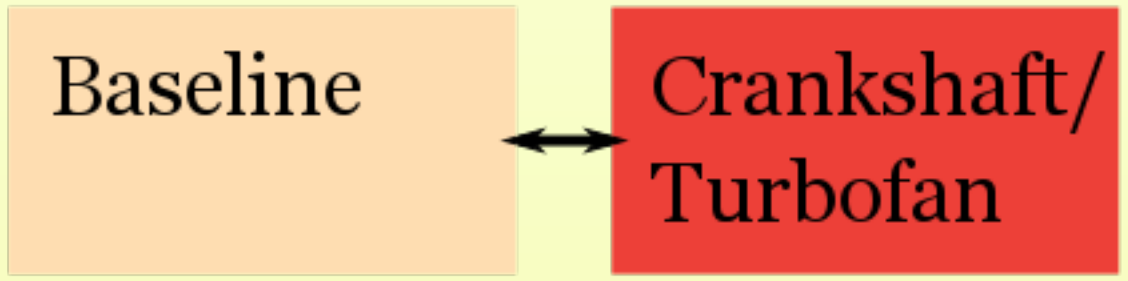
SpiderMonkey (Firefox)



JavaScriptCore (WebKit, Safari)



V8 (Chrome)



V8 perf

End of the road for Crankshaft

New thing: Turbofan

Fully typed IR, more capable of reliably inferring types over big asm.js programs

Sea-of-nodes approach transparently enables code motion

Status: enabled, but for asm.js code only

Features

ECMAScript 6 (ES6) was supposed to arrive this year, punted to next year, but all implementors involved in process

All engines are actively implementing ES6 features

JSC has implemented some features but not as focussed

`http://kangax.github.io/compat-table/es6/`

Desktop browsers												
6%	21%	72%	46%	59%	61%	69%	53%	56%	10%	10%	24%	28%
IE 10	IE 11	IE Technical Preview[1]	FF 31	FF 34	FF 35	FF 36	CH 39, OP 26[2]	CH 40, OP 27[2]	SF 6	SF 7.0	SF 7.1, SF 8	WK
No	No	No	No	No	No	No	No	No	No	No	No	No
0/9	0/9	8/9	7/9	7/9	7/9	7/9	4/9	4/9	0/9	0/9	0/9	0/9
0/8	8/8	8/8	3/8	3/8	3/8	8/8	5/8	5/8	1/8	1/8	1/8	1/8
0/10	8/10	8/10	0/10	0/10	0/10	0/10	5/10	5/10	0/10	0/10	0/10	0/10
0/5	0/5	0/5	3/5	3/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	0/5
No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No
0/8	0/8	4/8	6/8	6/8	6/8	8/8	0/8	0/8	0/8	0/8	2/8	2/8
0/8	0/8	8/8	0/8	0/8	0/8	0/8	0/8	4/8	0/8	0/8	0/8	0/8
0/3	0/3	3/3	0/3	0/3	0/3	0/3	0/3	1/3	0/3	0/3	0/3	0/3
0/3	0/3	3/3	0/3	3/3	3/3	3/3	1/3	2/3	0/3	0/3	1/3	1/3
0/4	0/4	4/4	3/4	3/4	3/4	4/4	4/4	4/4	0/4	0/4	1/4	1/4
0/8	0/8	0/8	6/8	6/8	7/8	8/8	8/8	8/8	0/8	0/8	0/8	0/8
0/4	0/4	2/4	2/4	2/4	2/4	4/4	4/4	4/4	0/4	0/4	0/4	0/4
0/2	0/2	1/2	0/2	2/2	2/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2
0/2	0/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	0/2
16/40	16/40	40/40	18/40	19/40	19/40	19/40	21/40	22/40	18/40	18/40	18/40	18/40
0/11	5/11	11/11	10/11	11/11	11/11	11/11	11/11	11/11	0/11	0/11	9/11	9/11
0/11	5/11	11/11	10/11	11/11	11/11	11/11	11/11	11/11	0/11	0/11	9/11	9/11
0/4	2/4	4/4	2/4	3/4	3/4	4/4	4/4	4/4	0/4	0/4	3/4	3/4
0/4	0/4	4/4	0/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	0/4	0/4
0/20	0/20	17/20	12/20	14/20	14/20	14/20	0/20	0/20	0/20	0/20	0/20	0/20
0/14	0/14	13/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14
No	Yes	Yes	No	No	No	No	Yes	Yes	No	No	No	No
0/11	0/11	0/11	5/11	7/11	7/11	8/11	0/11	0/11	0/11	0/11	5/11	5/11
No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
0/4	1/4	3/4	2/4	3/4	3/4	4/4	3/4	3/4	0/4	0/4	0/4	0/4
0/16	0/16	0/16	3/16	4/16	4/16	4/16	3/16	3/16	3/16	3/16	3/16	3/16
No	No	No	No	No	No	No	Yes	Yes	No	No	No	No
0/2	0/2	2/2	1/2	2/2	2/2	2/2	1/2	1/2	0/2	0/2	0/2	0/2
0/6	0/6	5/6	5/6	5/6	5/6	5/6	5/6	5/6	0/6	0/6	0/6	3/6
No	No	Yes	No	No	No	No	No	No	No	No	No	No
0/9	0/9	8/9	0/9	0/9	0/9	9/9	8/9	8/9	0/9	0/9	0/9	0/9
0/7	0/7	2/7	0/7	0/7	0/7	1/7	2/7	3/7	0/7	0/7	0/7	0/7
0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5
0/2	0/2	2/2	1/2	2/2	2/2	2/2	1/2	1/2	0/2	0/2	0/2	0/2

Trends

Architectural convergence

Ongoing perf work to make JS a better language to compile to (OdinMonkey, FTL, TF)

Ongoing ES6 feature work to make JS a better language to write

2015 predictions

TF landing?

More LLVM passes enabled in FTL?

ES6, ES7, other language experimentations?

?