Windows Port: WebAssembly and FTL JIT

Ian Grunert, Recurse Center Fall 1 Batch

Why?

Short-term:

Playwright, Bun

Port alignment

Longer term:

Cross-platform desktop apps

General purpose browsers

Signal handlers on Windows (259108)

Required for WebAssembly fast memory

Implemented on Windows using Vectored Exception Handling

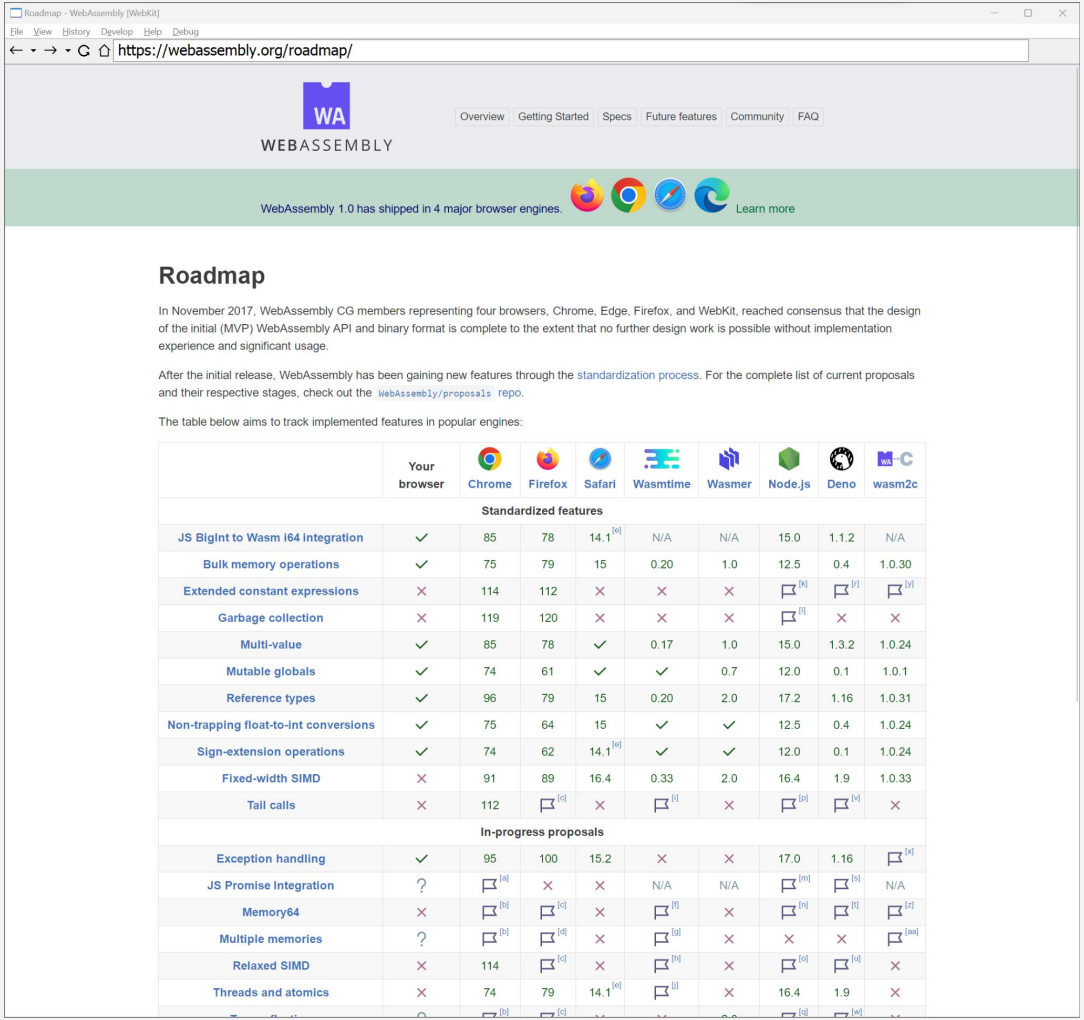
Landed!

Enable WebAssembly on Windows (222315)

Specifically, the Low Level Interpreter

Workaround in place for offlineasm loading global labels from WTF (175104)

PR is open, should land shortly



Roadmap

In November 2017, WebAssembly CG members representing four browsers, Chrome, Edge, Firefox, and WebKit, reached consensus that the design of the initial (MVP) WebAssembly API and binary format is complete to the extent that no further design work is possible without implementation experience and significant usage.

After the initial release, WebAssembly has been gaining new features through the standardization process. For the complete list of current proposals and their respective stages, check out the `WebAssembly/proposals` repo.

The table below aims to track implemented features in popular engines:

	Your browser	Chrome	Firefox	Safari	Wasmtime	Wasmer	Node.js	Deno	wasm2c
Standardized features									
JS BigInt to Wasm i64 integration	✓	85	78	14.1[e]	N/A	N/A	15.0	1.1.2	N/A
Bulk memory operations	✓	75	79	15	0.20	1.0	12.5	0.4	1.0.30
Extended constant expressions	✗	114	112	✗	✗	✗	[k]	[r]	[v]
Garbage collection	✗	119	120	✗	✗	✗	[l]	✗	✗
Multi-value	✓	85	78	✓	0.17	1.0	15.0	1.3.2	1.0.24
Mutable globals	✓	74	61	✓	✓	0.7	12.0	0.1	1.0.1
Reference types	✓	96	79	15	0.20	2.0	17.2	1.16	1.0.31
Non-trapping float-to-int conversions	✓	75	64	15	✓	✓	12.5	0.4	1.0.24
Sign-extension operations	✓	74	62	14.1[e]	✓	✓	12.0	0.1	1.0.24
Fixed-width SIMD	✗	91	89	16.4	0.33	2.0	16.4	1.9	1.0.33
Tail calls	✗	112	[c]	✗	[i]	✗	[p]	[v]	✗
In-progress proposals									
Exception handling	✓	95	100	15.2	✗	✗	17.0	1.16	[x]
JS Promise Integration	?	[a]	✗	✗	N/A	N/A	[m]	[s]	N/A
Memory64	✗	[b]	[c]	✗	[f]	✗	[n]	[t]	[z]
Multiple memories	?	[b]	[d]	✗	[g]	✗	✗	✗	[aa]
Relaxed SIMD	✗	114	[c]	✗	[h]	✗	[o]	[u]	✗
Threads and atomics	✗	74	79	14.1[e]	[j]	✗	16.4	1.9	✗
[illegible]		[b]	[c]				[q]	[w]	

Enable FTL JIT on Windows (145366)

FTL assumes WebAssembly is built (even if runtime disabled)

ARES-6 is working

Known issues:

DFG operations returning UGPRPair

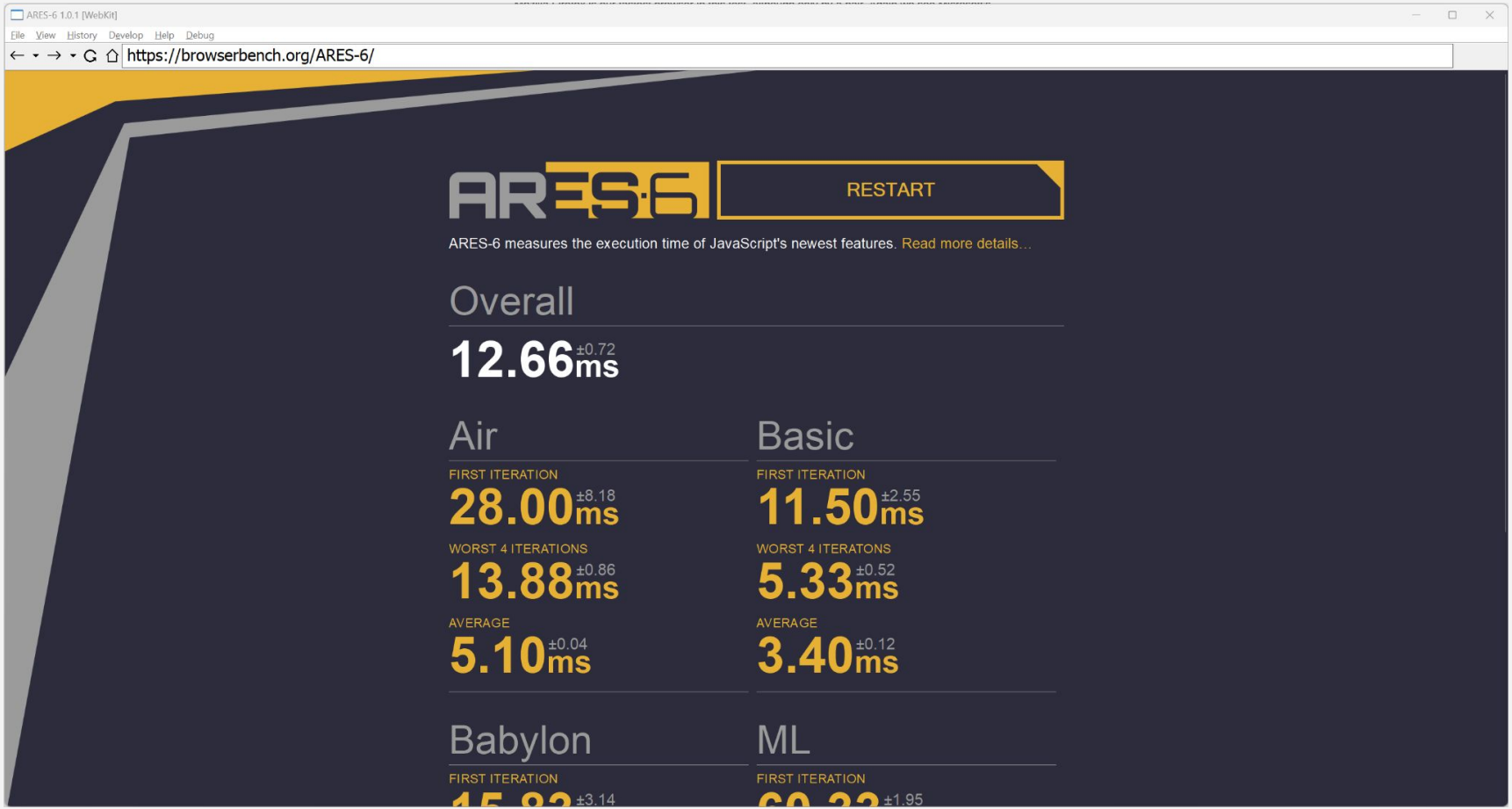
Fails during JetStream2 when running stanford-crypto-aes

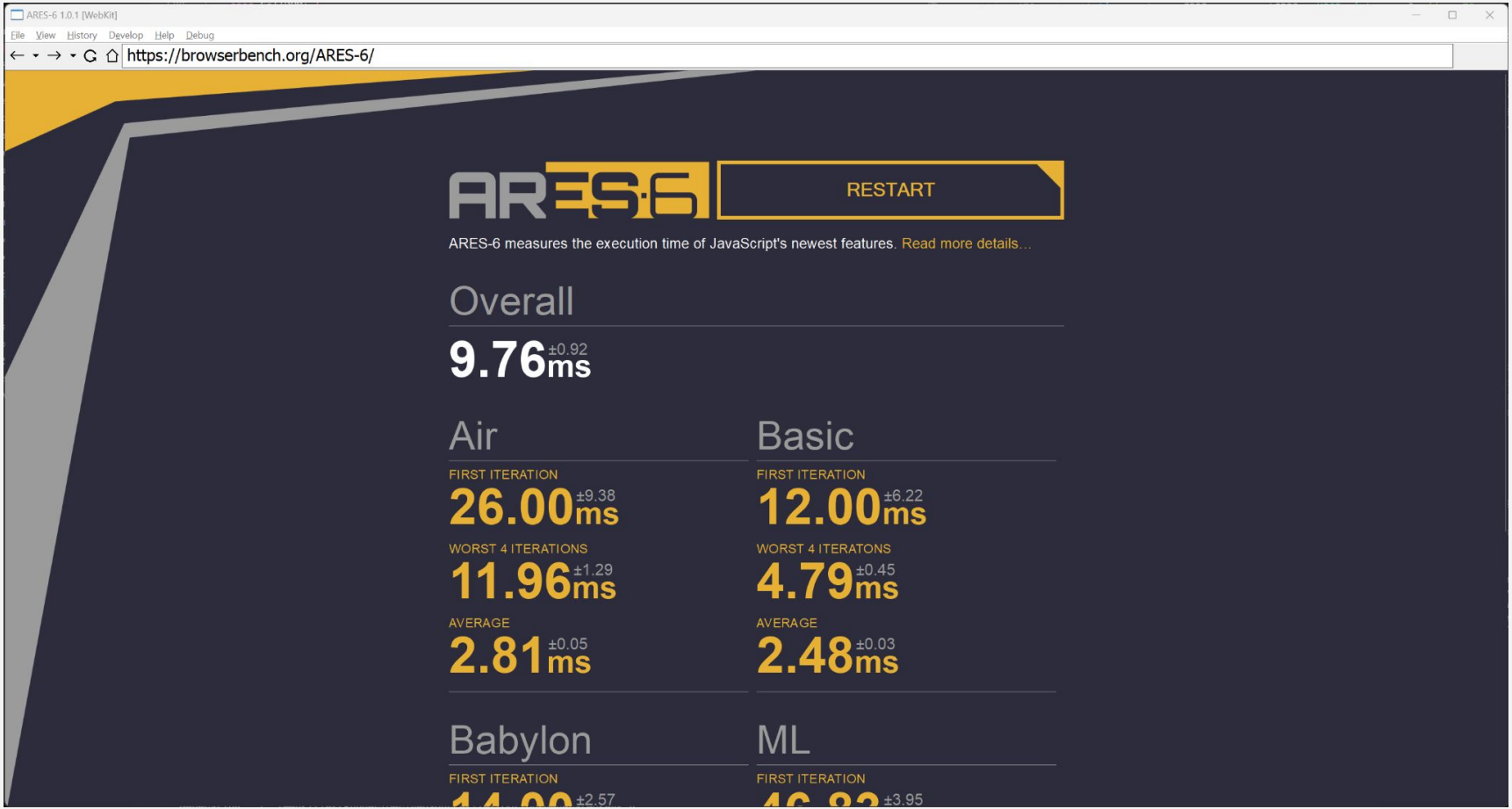
Bonus: WebAssembly BBQ and OMG JIT

WebAssembly JITs enable with FTL

BBQ JIT fails on Windows when running JetStream 2

Runs out of GPRs to assign as it doesn’t use callee-saves


ARES-6 1.0.1 [WebKit]
File View History Develop Help Debug
https://browserbench.org/ARES-6/
ARES-6
RESTART
ARES-6 measures the execution time of JavaScript's newest features. Read more details...
Overall
12.66 ±0.72 ms
Air
FIRST ITERATION
28.00 ±8.18 ms
WORST 4 ITERATIONS
13.88 ±0.86 ms
AVERAGE
5.10 ±0.04 ms
Basic
FIRST ITERATION
11.50 ±2.55 ms
WORST 4 ITERATONS
5.33 ±0.52 ms
AVERAGE
3.40 ±0.12 ms
Babylon
FIRST ITERATION
±3.14
ML
FIRST ITERATION
±1.95


ARES-6 1.0.1 [WebKit]
File View History Develop Help Debug
https://browserbench.org/ARES-6/
ARES-6
RESTART
ARES-6 measures the execution time of JavaScript's newest features. Read more details...
Overall
9.76 ±0.92 ms
Air
FIRST ITERATION
26.00 ±9.38 ms
WORST 4 ITERATIONS
11.96 ±1.29 ms
AVERAGE
2.81 ±0.05 ms
Basic
FIRST ITERATION
12.00 ±6.22 ms
WORST 4 ITERATONS
4.79 ±0.45 ms
AVERAGE
2.48 ±0.03 ms
Babylon
FIRST ITERATION
ML
FIRST ITERATION
±3.95

What’s next?

Land work in progress

Callee-save support for BBQ JIT

Static asserts in offlineasm (203692)

Performance - Speedometer

WebAssembly In Place Interpreter

bmalloc / libpas

Cross-compilation?

Thanks!