An ahead-of-time JavaScript compiler

Porffor compiles JavaScript ahead-of-time to WebAssembly and native binaries.
It is currently in pre-alpha with usability beginning in 2025.

JS → Wasm

Porffor's WebAssembly output is 10-30x smaller and faster compared to existing JS → Wasm projects as Porffor compiles JS instead of bundling an interpreter in the Wasm output.

JS as Wasm allows for sandboxed execution but suffers drastic performance losses: Porffor solves this, allowing for:

- Efficient, secure server-side JS hosting
- Reverse-engineer resistance

JS → Native

As Porffor truly compiles JS without packaging a runtime like existing solutions, binary sizes are 1000x smaller (~90MB → <100KB) with potential other performance benefits too.

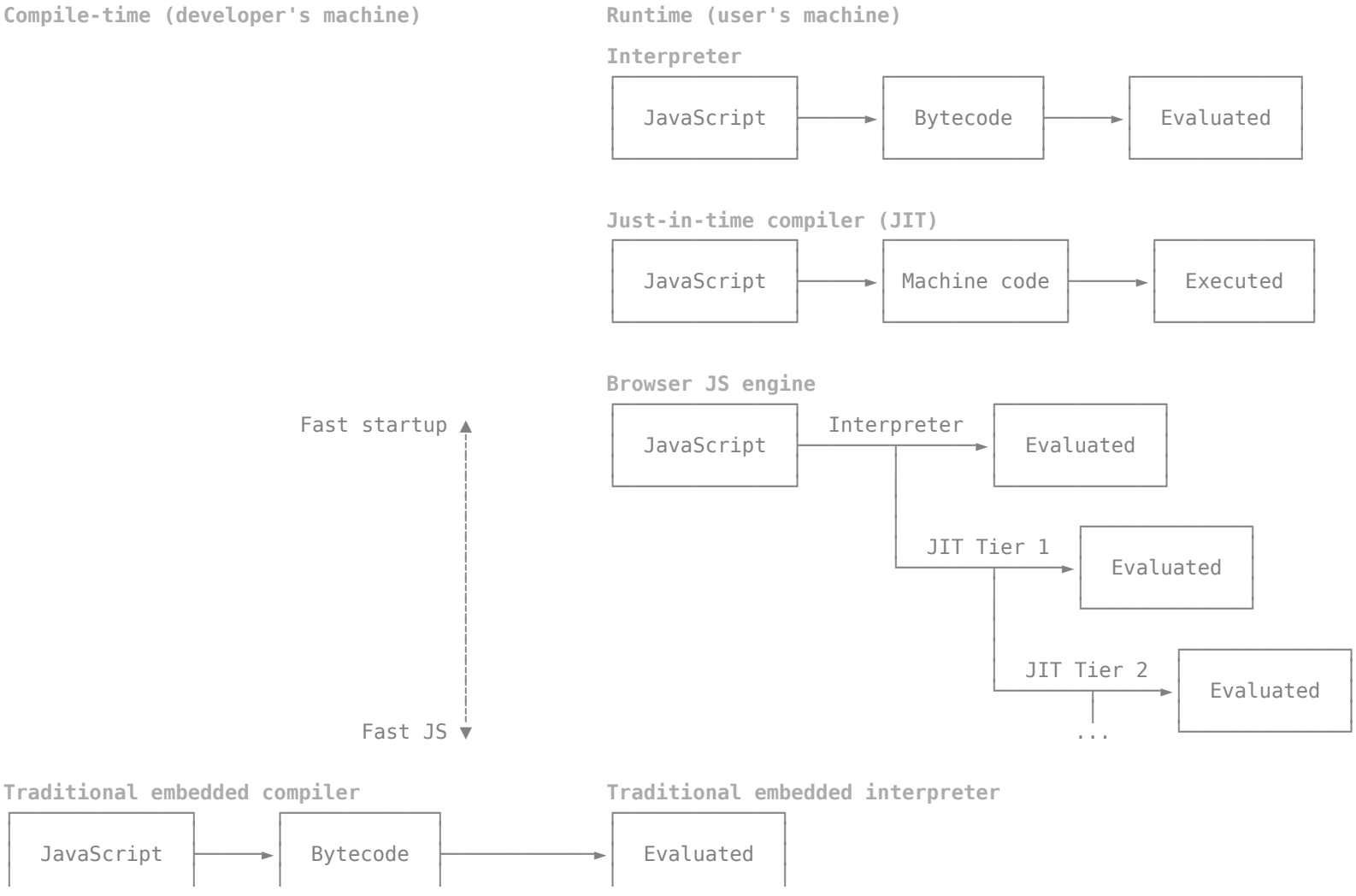
JS as native allows for simple one-click execution but has substandard sizes: **Porffor solves this**, allowing for:

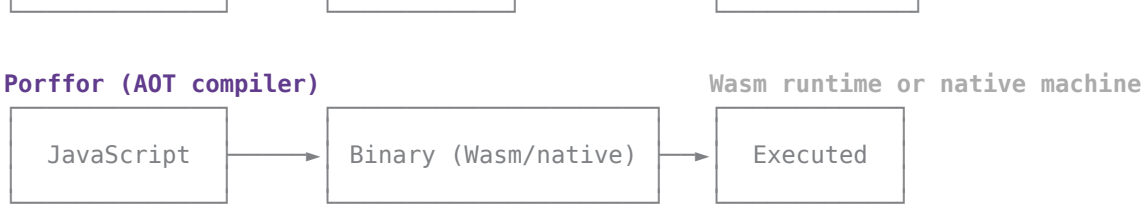
- **Fast JS for embedded, game consoles, and more**
- **Tiny CLI apps written in JS**

Porffor has safe compilation as it is written in JS (preventing memory safety vulnerabilities) with 0 eval.

Porffor is written from scratch with ahead-of-time in mind allowing for previously impossible optimizations.

Porffor natively supports TypeScript, no transpilation step (even internally): just give it a TS file.





Instead of a slow interpreter or multiple JIT tiers carefully balancing compilation speed (startup/loading time) and JS performance, compiling AOT allows you to compile first and run later. While compile speed for AOT remains important for DX, it no longer effects UX; allowing Porffor to perform static analysis for optimizations like C++ or Rust do today.

The main drawback of AOT is that runtime JS execution (eval, new Function, etc) is infeasible. Also as Porffor is early, it is unstable and not all JS works yet.

Test262

Porffor is ran against Test262, the official ECMAScript conformance test suite, every commit to track conformance progress.