Profiles

Andreas Rossberg

Wasm CG 2025/10/29

Platform diversity is real!

and it will only increase over time

This is a good thing!

Consequence of Wasm's broad success!

Many different (and largely disjoint) eco-systems

…with different, incompatible constraints

	GraalWasm	Wabt	WAMR	Wasm3	WasmEdge	Wasmer	Wasmi	Wasmtime	WAVM	wazero	Wizard
reftypes	⏯	⏯	✅	❌	⏯	⏯	✅	⏯	⏯	⏯	✅
SIMD	❌	⏯	✅	❌	⏯	✅	❌	⏯	⏯	⏯	✅
threads	⏯	⏯	✅	❌	⏯	⏯	❌	⏯	⏯	❌	⏯

[2023]

✅ always on

❌ not provided

⏯ flag

GC, exceptions,
stack switching, func.new, …

Concrete example: Lime1

Lime1

The Lime1 target consists of WebAssembly 1.0 plus the following standardized (phase-5) features:

- mutable-globals
- multivalue
- sign-ext
- nontrapping-fptoint
- bulk-memory-opt
- extended-const
- call-indirect-overlong

This set of features can be tested for by using the Lime1 test. This is not a comprehensive conformance test, but it does at least minimally use all of the features mentioned above.

Feature subsets

WebAssembly features sometimes contain several features combined into a single proposal to simplify the standardization process, but can have very different implementation considerations. This section defines subsets of standardized features for use in Lime configurations.

bulk-memory-opt

bulk-memory-opt is a subset of the bulk-memory feature that contains just the `memory.copy` and `memory.fill` instructions.

It does not include the table instructions, `memory.init`, or `data.drop`.

call-indirect-overlong

call-indirect-overlong is a subset of the reference-types feature that contains just the change to the `call_indirect` instruction encoding to change the zero byte to an LEB encoding which may have an overlong encoding.

It does not include the actual reference types.

references 8 historical documents

modulo informal diffs

may miss interactions

Profiles

Precise specifications of sublanguages

Restrictions on syntax or semantics

Chosen by eco-systems, not applications!

Goals

Minimise fragmentation
in the face platform diversity & constraints

Maximise compatibility across similar ecosystems

Stable, durable, widely agreed subsets

Well-specified in a single place

Few and coarse

Non-Goals

Producer-side choice

Versioning

Feature detection

Alternate semantics

Intended Properties

Profiles need to be mutually compatible and composable

Producers should never assume *absence* of features

…targeted consumer may extend their profile over time

Deploy-time choice, avoid runtime conditionals on profiles

Approach

Subtractive not additive!

Spec coherently defines complete language

But carves out specific syntax/semantics rules to omit

Risk of Proactivity

Hard to predict requirements

…may introduce unneeded profiles

…may specify the wrong thing

Though for some proposals, it is easy to predict

Risk of Retroactivity

Too late for respective customers

…they are forced to make up a subset themselves

…high likely hood of incompatible choices

Customers have little incentive to go through separate proposal process

…too much hassle for too little benefit at that point

…takes too long

Consequence may be maximal fragmentation