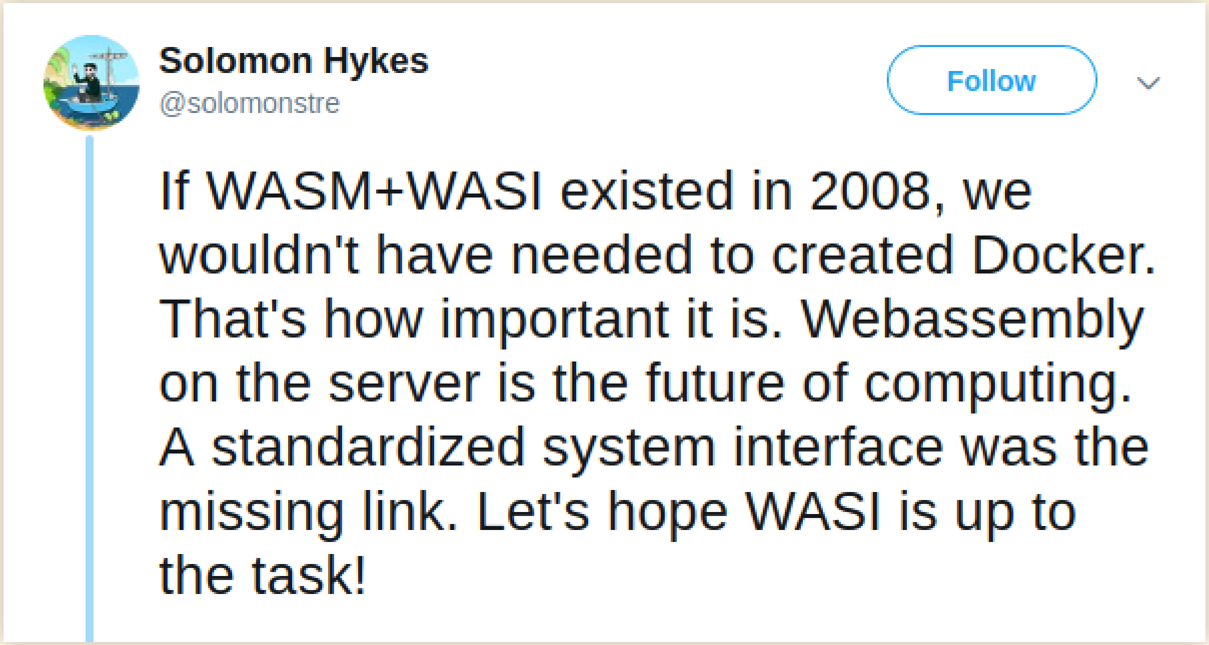
Server-Side C++ With WebAssembly

Disclaimer

- Not a WebAssembly expert

Why WebAssembly?

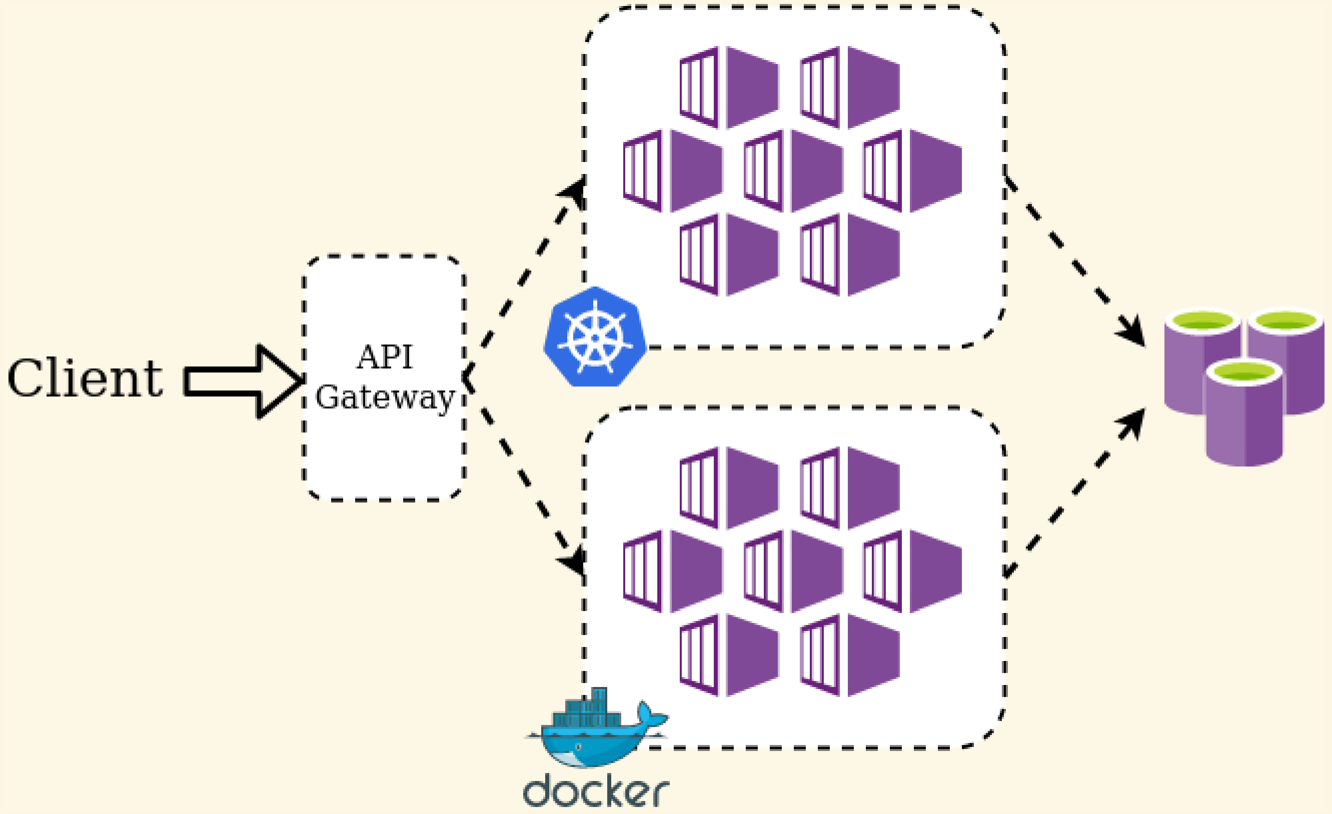


Solomon Hykes
@solomonstre

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

Solomon Hykes is the Founder, former CTO and Chief Architect of Docker.

MICROSERVICES
MICROSERVICES EVERYWHERE
LIGHTYEAR
SPACE RANGER


Client
API Gateway
docker

Java

- Slow startup times
- High RAM usage
- Unpredictable performance
- Large container images

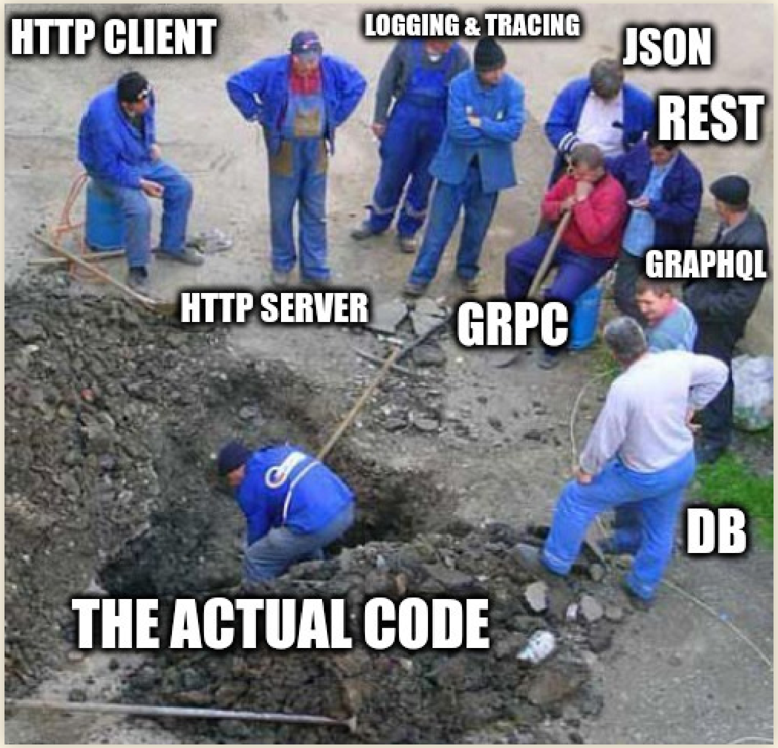
Go

- Also GC
- No const, enum, templates, RAII*, overloading etc.
- Very opinionated
- No standard, controlled by single corporation

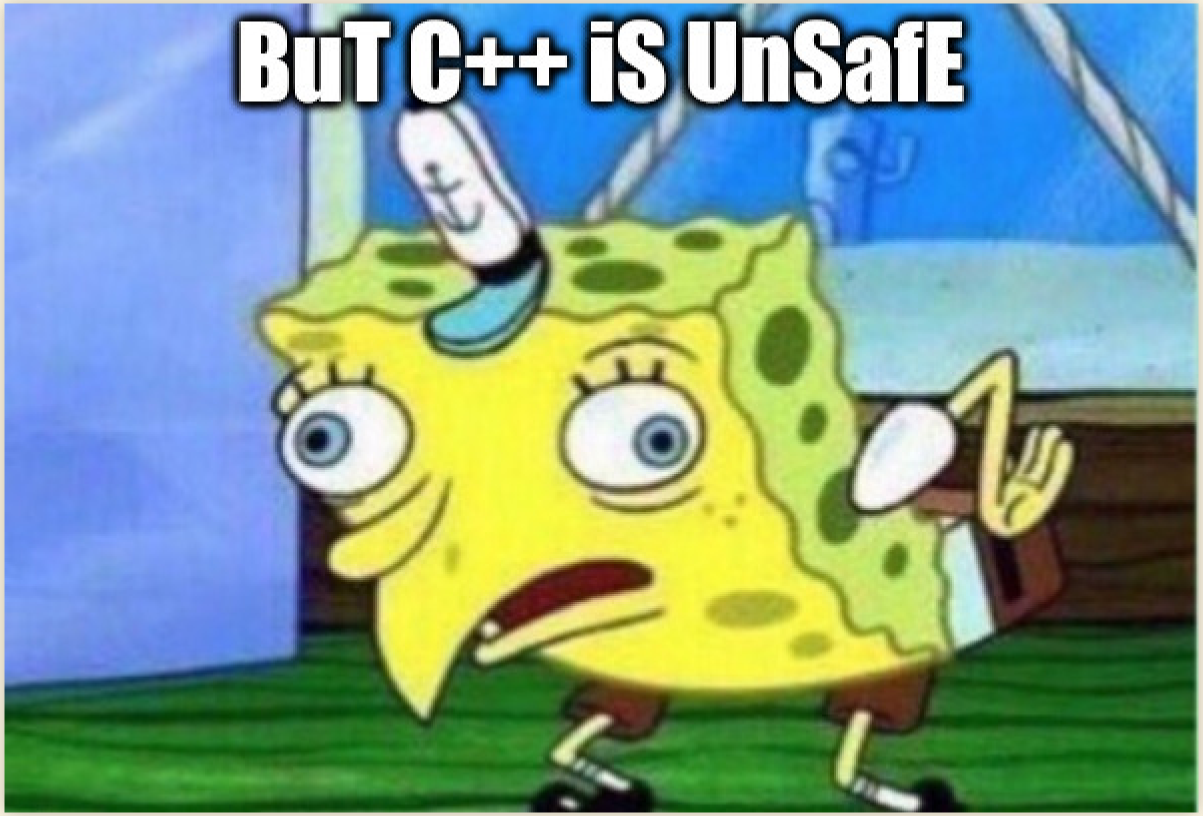
What About C++?

Lack of Ecosystem

- Complicated build systems (*make*, *ninja*, *cmake*, *autotools*)
- Fragmented package managers (*pkg-config*, *conan*, *vcpkg*, *build2*)
- Lack of packaged libraries



Safety



WebAssembly

*"Neither **web**, nor **assembly**"*

Everyone

JavaScript

JavaScript prototype developed at Netscape in **1995**

```
function factorial(n) {
    if (n === 0)
        return 1; // 0! = 1

    return n * factorial(n - 1);
}
```

asm.js

```
size_t strlen(char *ptr) {
  char *curr = ptr;
  while (*curr != 0) {
    curr++;
  }
  return (curr - ptr);
}
```

Emscripten

↓

```
function strlen(ptr) {
  ptr = ptr|0;
  var curr = 0;
  curr = ptr;
  while ((MEM8[curr>>0]|0) != 0)
    curr = (curr + 1)|0;
  return (curr - ptr)|0;
}
```

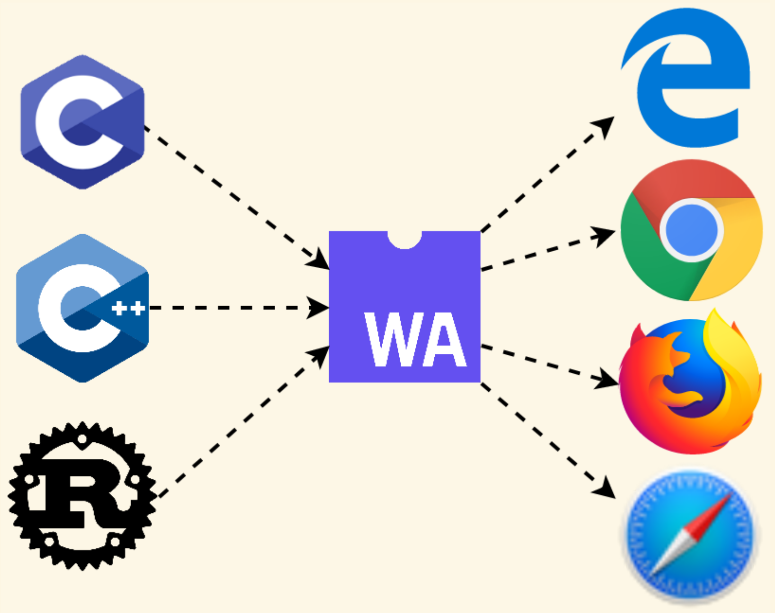
WebAssembly

"In a miracle of standards that never actually happens, everyone got together and agreed on something."

Steve Klabnik

Origin

- Originally designed by Mozilla, Microsoft, Google, and Apple
- Binary instruction format for a stack-based virtual machine
- Portable target for compilation of high-level languages



WebAssembly in the Browser Today

- W3C **standard** as of **December 5, 2019**
- Supported in **all** major browsers (including mobile)
- Unity, Unreal, Godot, Construct3
- Autocad, Google Earth, VLC
- Qt, SDL

TEACHING C++ TO BEGINNERS
Why would you do something so controversial, yet so brave
Cppcon The C++ Conference
2019 cppcon.org
Ben Smith
Applied WebAssembly: Compiling and Running C++ in Your Web Browser
Video Sponsorship Provided By: ansatz

Ben Smith: "Applied WebAssembly: Compiling and Running C++ in Your Web Browser"

So What Exactly Is WebAssembly?

WebAssembly Specification

Release 1.0

WebAssembly Community Group

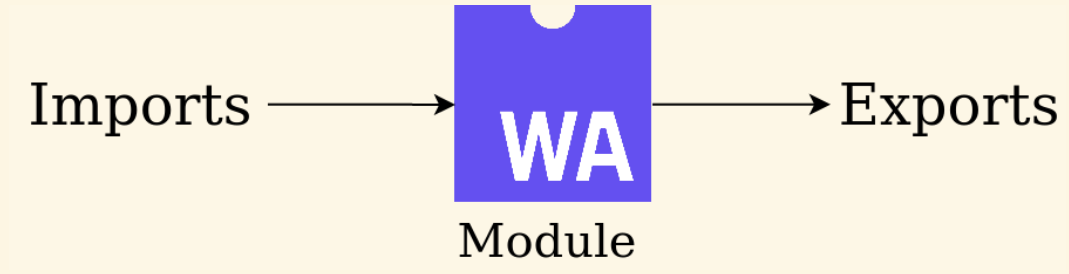
Andreas Rossberg (editor)

WebAssembly Specification

- Module format
- Virtual machine
- Instruction set
- Binary and text encoding

Module

Distributable, loadable, and executable unit of code

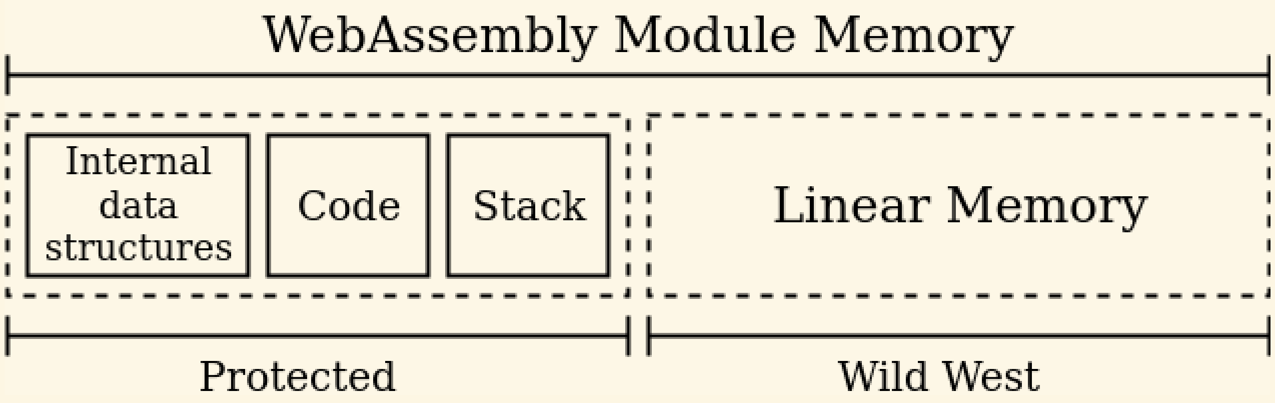


Virtual Machine

- 32-bit typed stack machine
- Value Types (little endian): `i32`, `i64`, `f32`, `f64`
- Code is validated before execution
- Machine verified type system

Memory

- Harvard architecture
- All memory accesses are bounds checked



Structured Control Flow

- No arbitrary jumps/goto
- `block`, `loop`, `if`, `br`

Text Format

Linear

```
(func $sum (type 0) (param i32 i32) (result i32)
    local.get 1
    local.get 0
    i32.add)
```

Folded (s-expressions)

```
(func $sum (type 0) (param i32 i32) (result i32)
    (i32.add
      (local.get 1)
      (local.get 0)))
```

WebAssembly Execution Example

`sum(2, 3)`

Function code	Bin	Stack
`(func $sum (param i32 i32) (result i32)`		
`    local.get 0`	`20 00`	
`    local.get 1`	`20 01`	
`    i32.add`	`6a`	
`)`	`0b`	

Locals:

```
[]
```

Reset | Next

What About C++?

What Works

- Everything that happens at compile time
- Stack access (*aliased stack*)
- Arbitrary control flow (*relooper algorithm*)
- Function pointers, vtable (`call_indirect`)
- RTTI, dynamic_cast, noexcept*, varargs

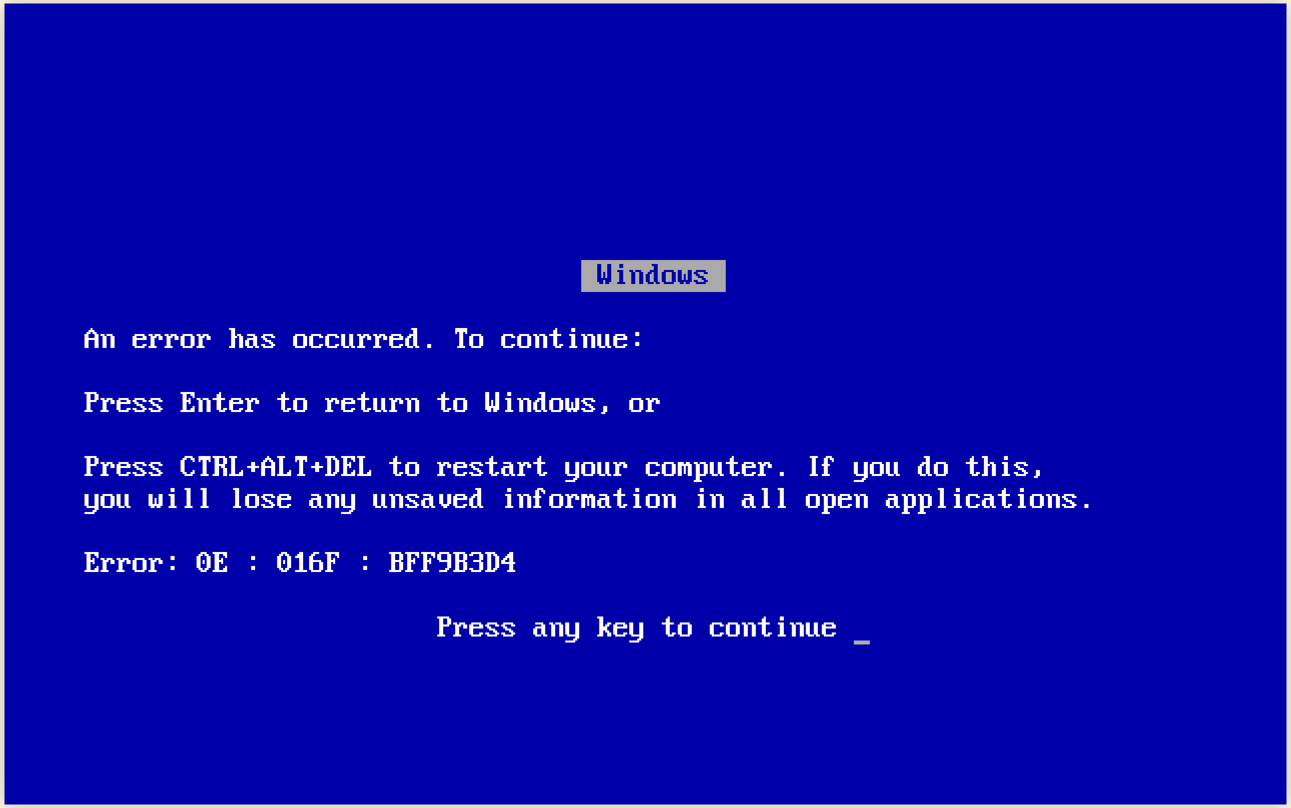
What Doesn't Work

- Threads, atomic operations (threads proposal)
- SIMD (SIMD proposal)
- Exceptions (exceptions proposal)
- Dynamic initialization, `atexit`
- System calls (WASI)

Compiling C++ to WASM

- Emscripten
- Clang 8+

Live Demo!



```
                         Windows

An error has occurred. To continue:

Press Enter to return to Windows, or

Press CTRL+ALT+DEL to restart your computer. If you do this,
you will lose any unsaved information in all open applications.

Error: 0E : 016F : BFF9B3D4

                      Press any key to continue _
```

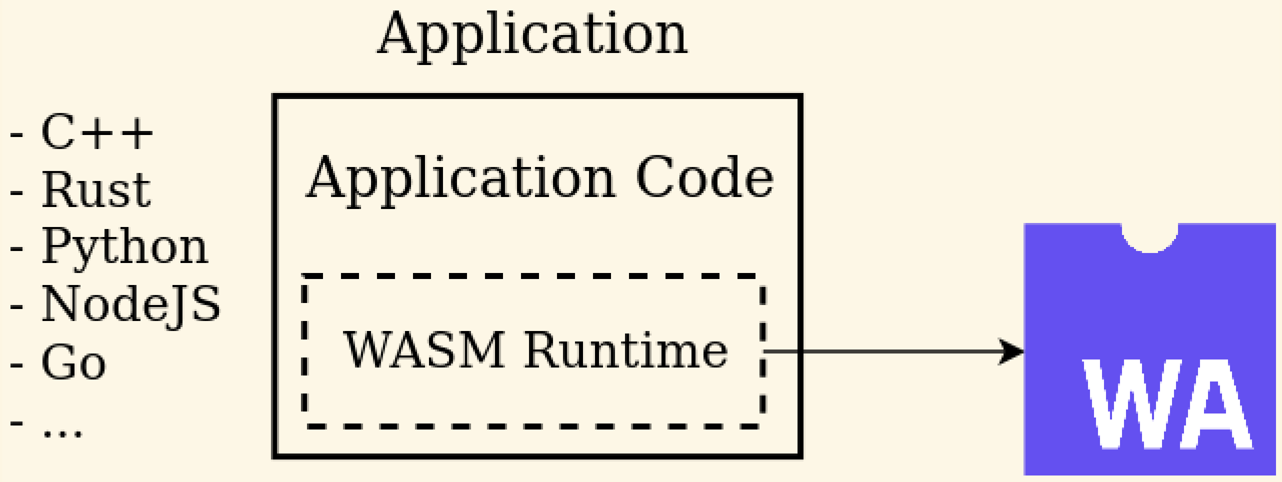
Why Outside the Browser?

- Safety and security
- Isolation
- Portability
- Performance
- Standard

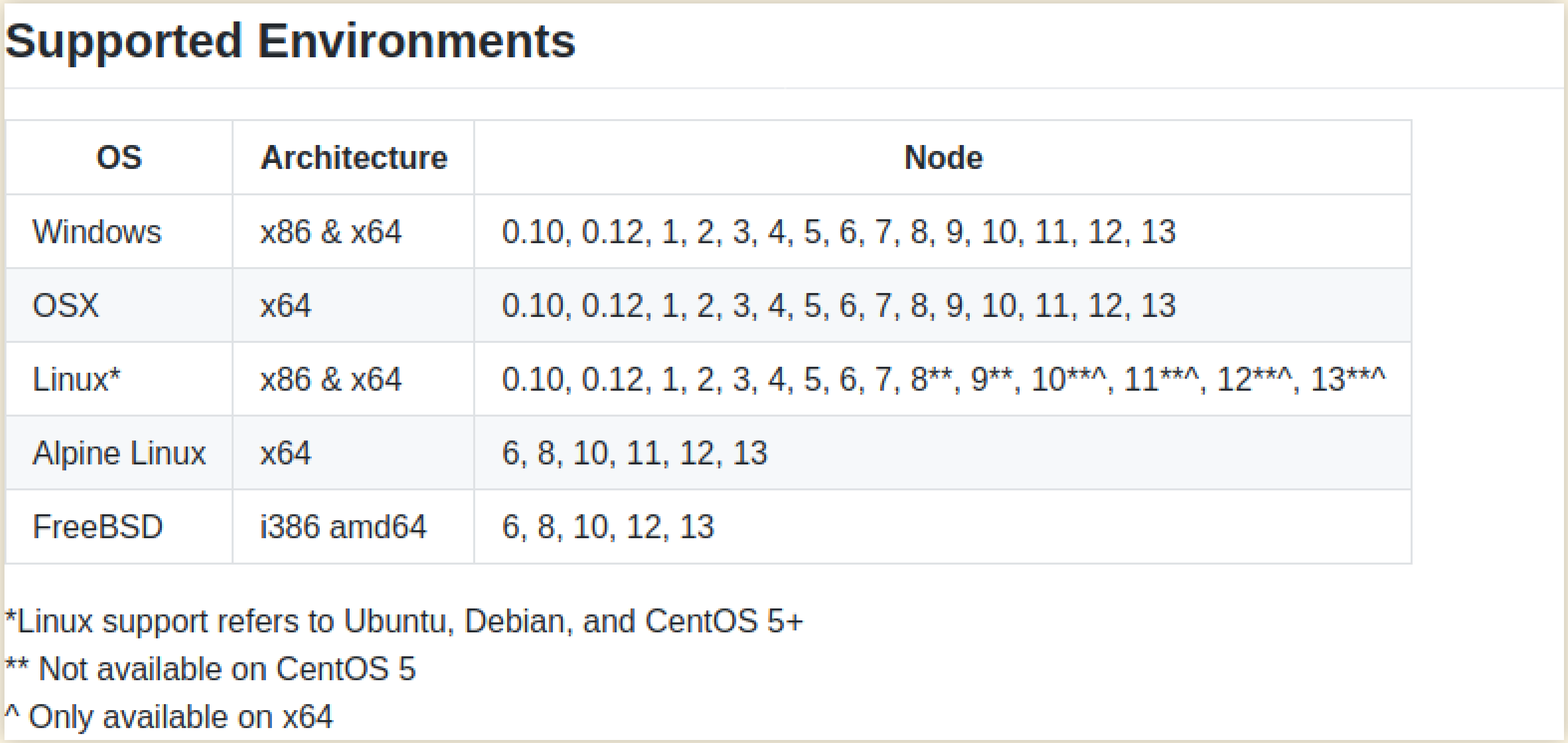
Standalone WebAssembly Runtimes

- Wasmtime (Mozilla)
- Lucet (Fastly)
- Node.js (V8)
- WAMR, Wasmer, WAVM, WAC...

Embedding Runtime



node-sass

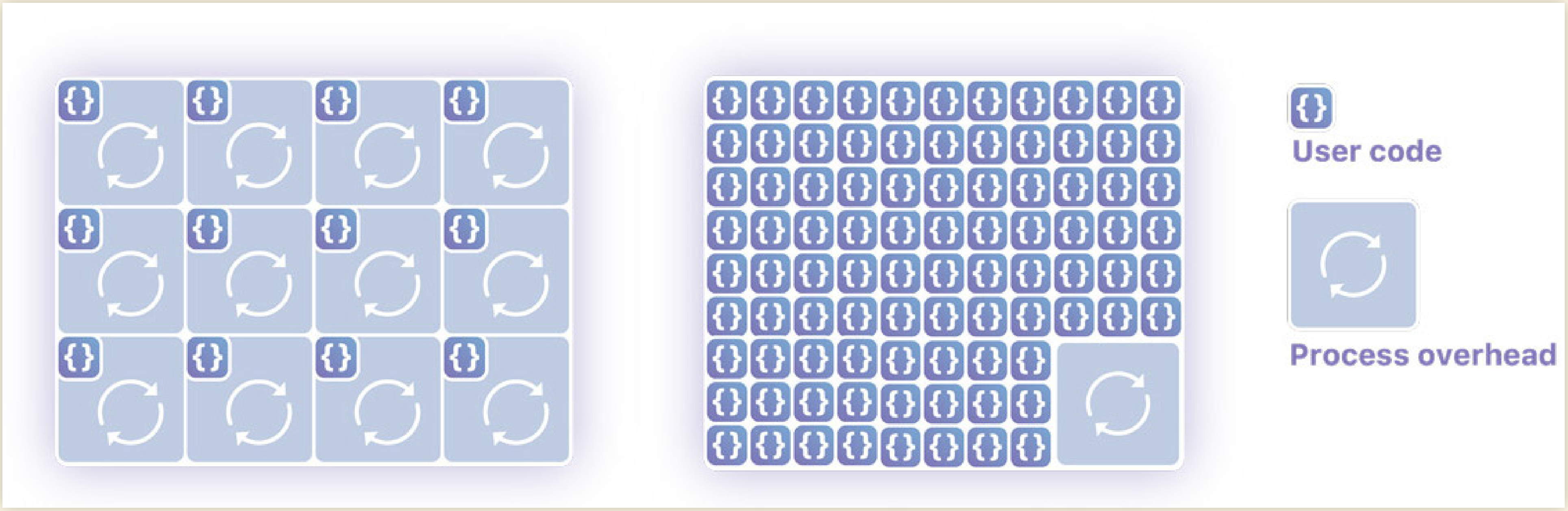


Supported Environments

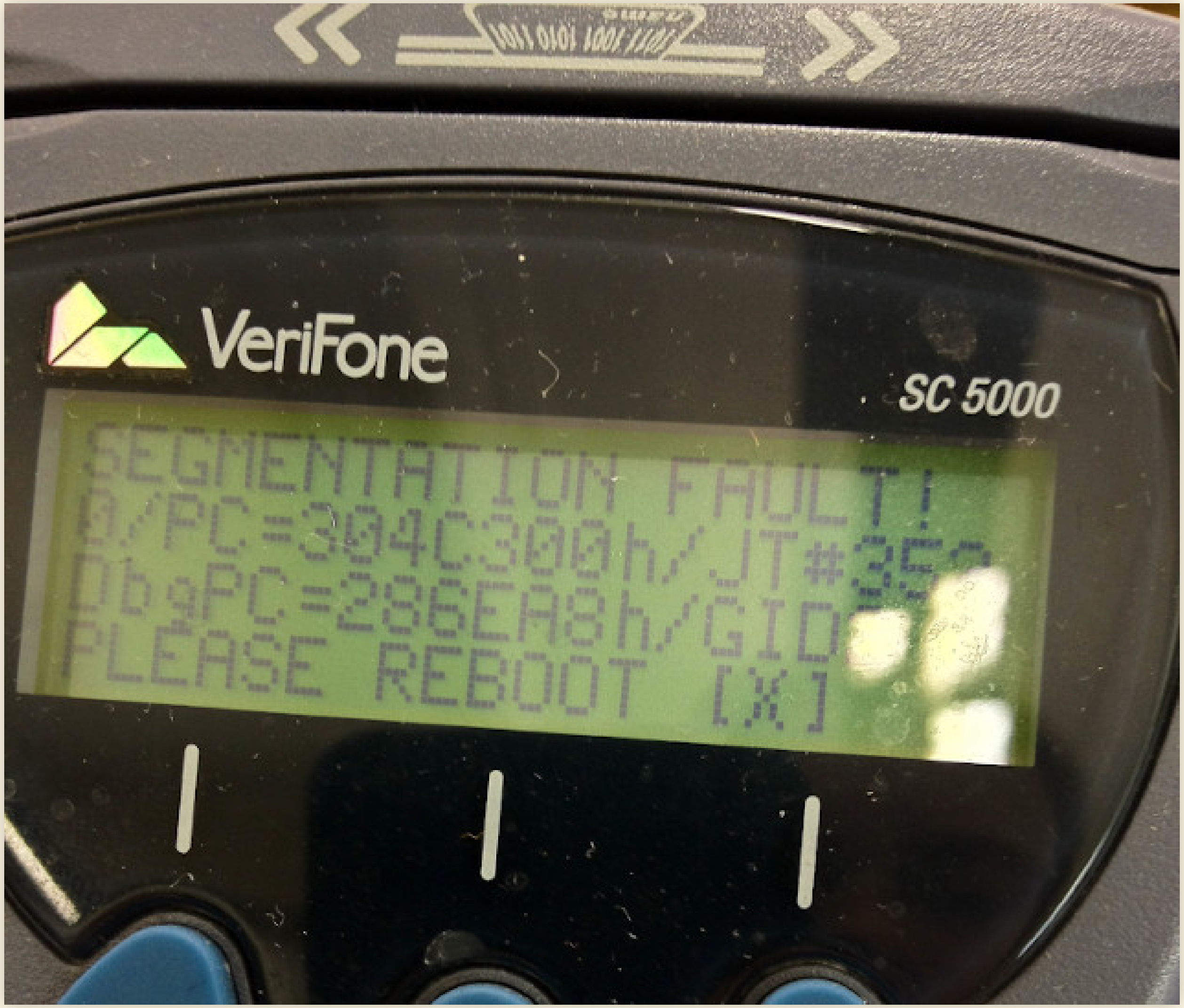
OS	Architecture	Node
Windows	x86 & x64	0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
OSX	x64	0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
Linux*	x86 & x64	0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8**, 9**, 10**^, 11**^, 12**^, 13**^
Alpine Linux	x64	6, 8, 10, 11, 12, 13
FreeBSD	i386 amd64	6, 8, 10, 12, 13

*Linux support refers to Ubuntu, Debian, and CentOS 5+
** Not available on CentOS 5
^ Only available on x64

Serverless



Live Demo!



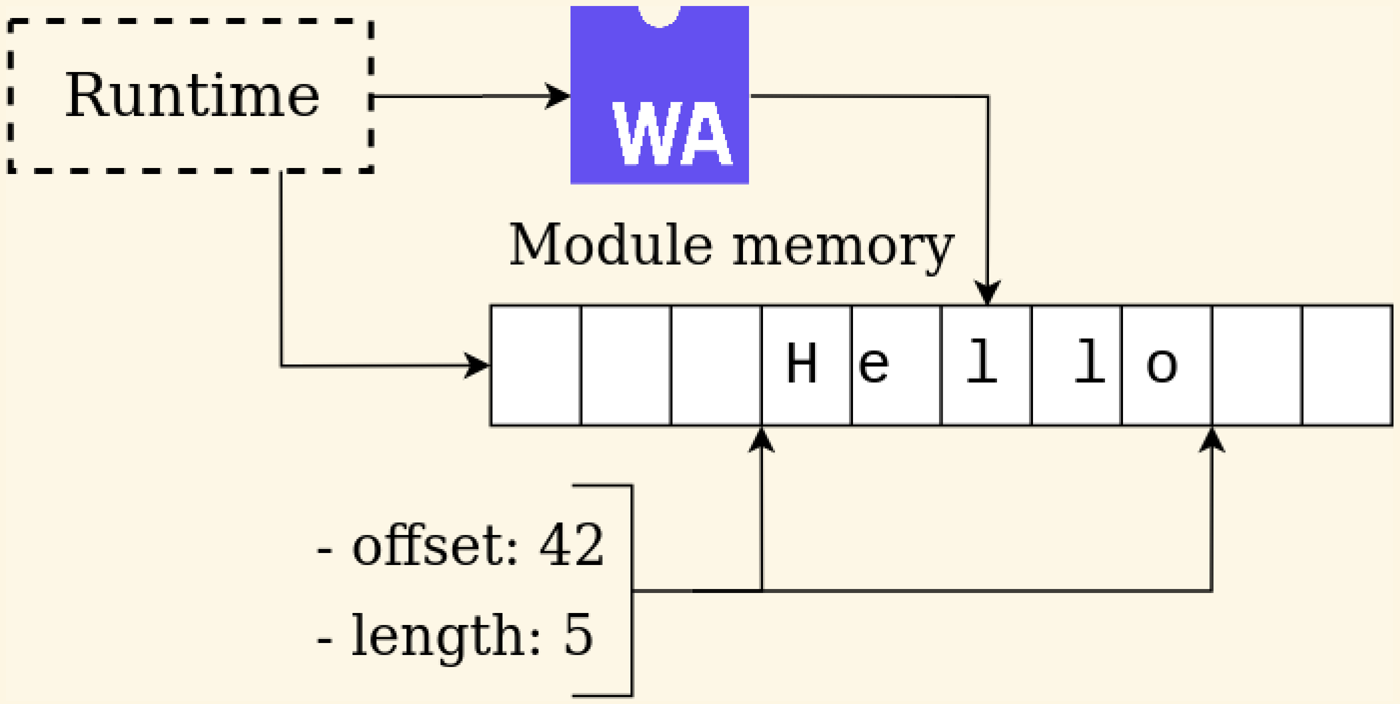
```
SEGMENTATION FAULT!
0/PC=304C300h/JT#35
DbgPC=286EA8h/GID
PLEASE REBOOT [X]
```

Interfacing With WebAssembly

Numbers in, numbers out

```
(func sum (param i32 i32) (result i32)
    local.get 1
    local.get 0
    i32.add)
```

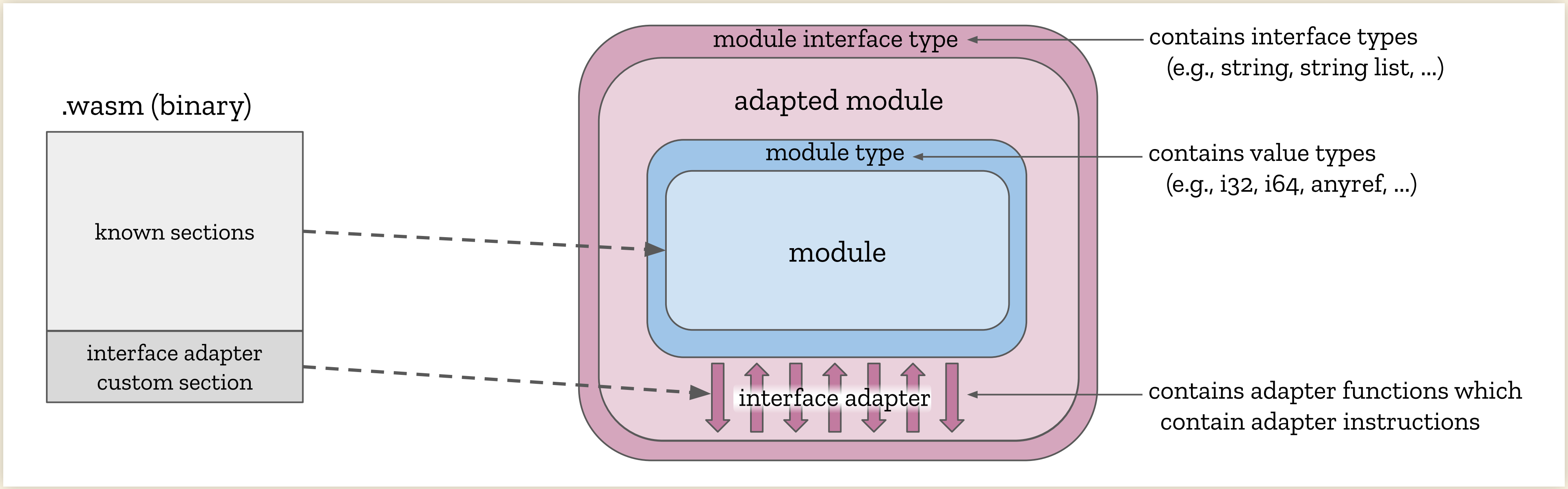
Passing/Returning a String



Not an Ideal Solution

- Lot of tedious work
- Manual memory management
- Multiple type representations

Interface Types


.wasm (binary)
known sections
interface adapter custom section
module interface type
adapted module
module type
module
interface adapter
contains interface types (e.g., string, string list, ...)
contains value types (e.g., i32, i64, anyref, ...)
contains adapter functions which contain adapter instructions

Returning a String

```
(module
  (memory (export "mem") 1)
  (data (i32.const 0) "hello there")
  (func (export "greeting_") (result i32 i32)
    i32.const 0   ;; offset of string in memory
    i32.const 11  ;; length
  )
)
```

Returning a String

```
(module
  (memory (export "mem") 1)
  (data (i32.const 0) "hello there")
  (func (export "greeting_") (result i32 i32)
    i32.const 0     ;; offset of string in memory
    i32.const 11    ;; length
  )
  (@interface func (export "greeting") (result string)
    call-export "greeting_" ;; call greeting_
    memory-to-string "mem"  ;; offset+length -> string
  )
)
```

Dynamically Allocated String

```
(@interface func (export "greeting") (result string)
    call-export "greeting_"
    defer-call-export "free"
    memory-to-string "mem"
  )
```

Live Demo!

Error 404: Demo Not Found

Interfacing With the System

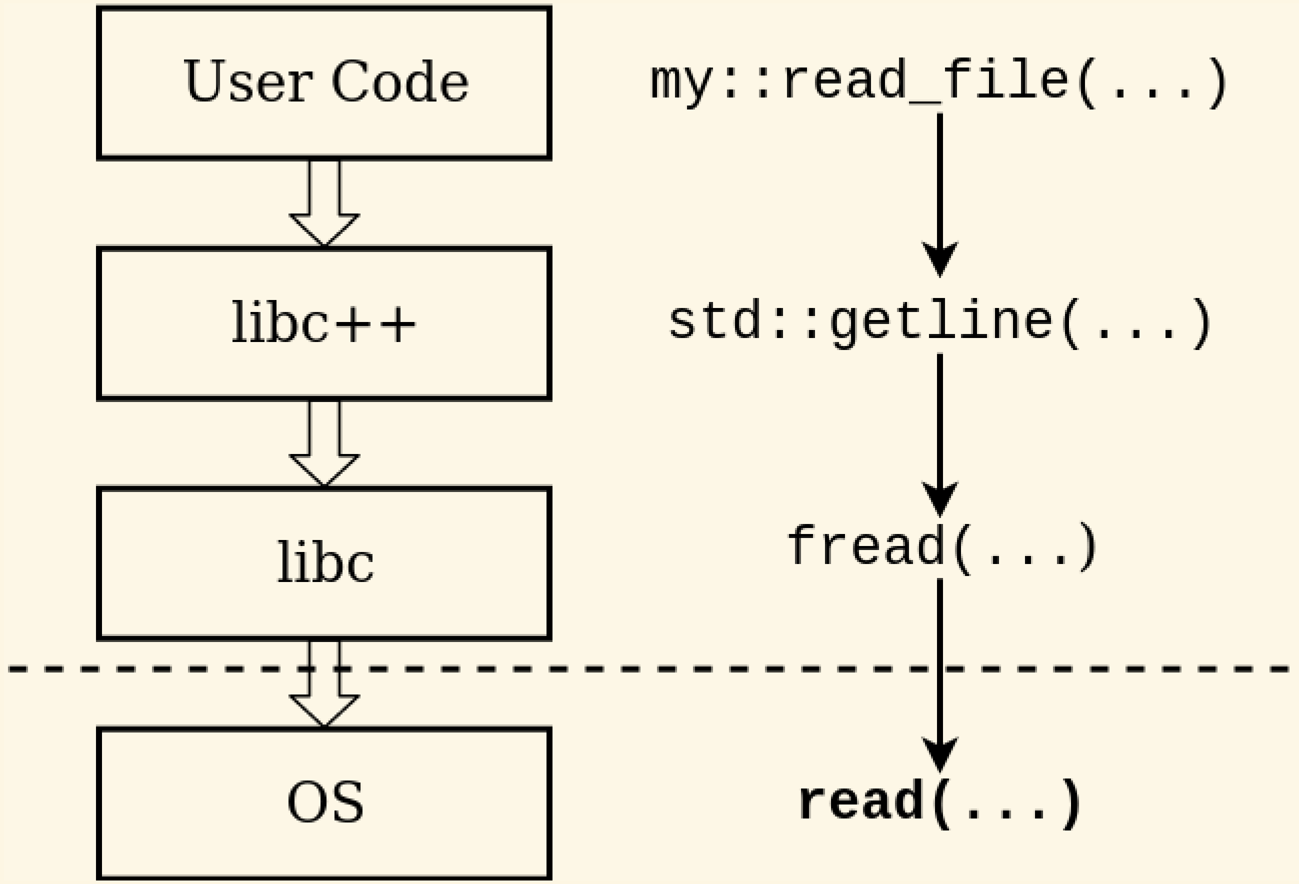
“WebAssembly cannot do anything.”

Ben Smith

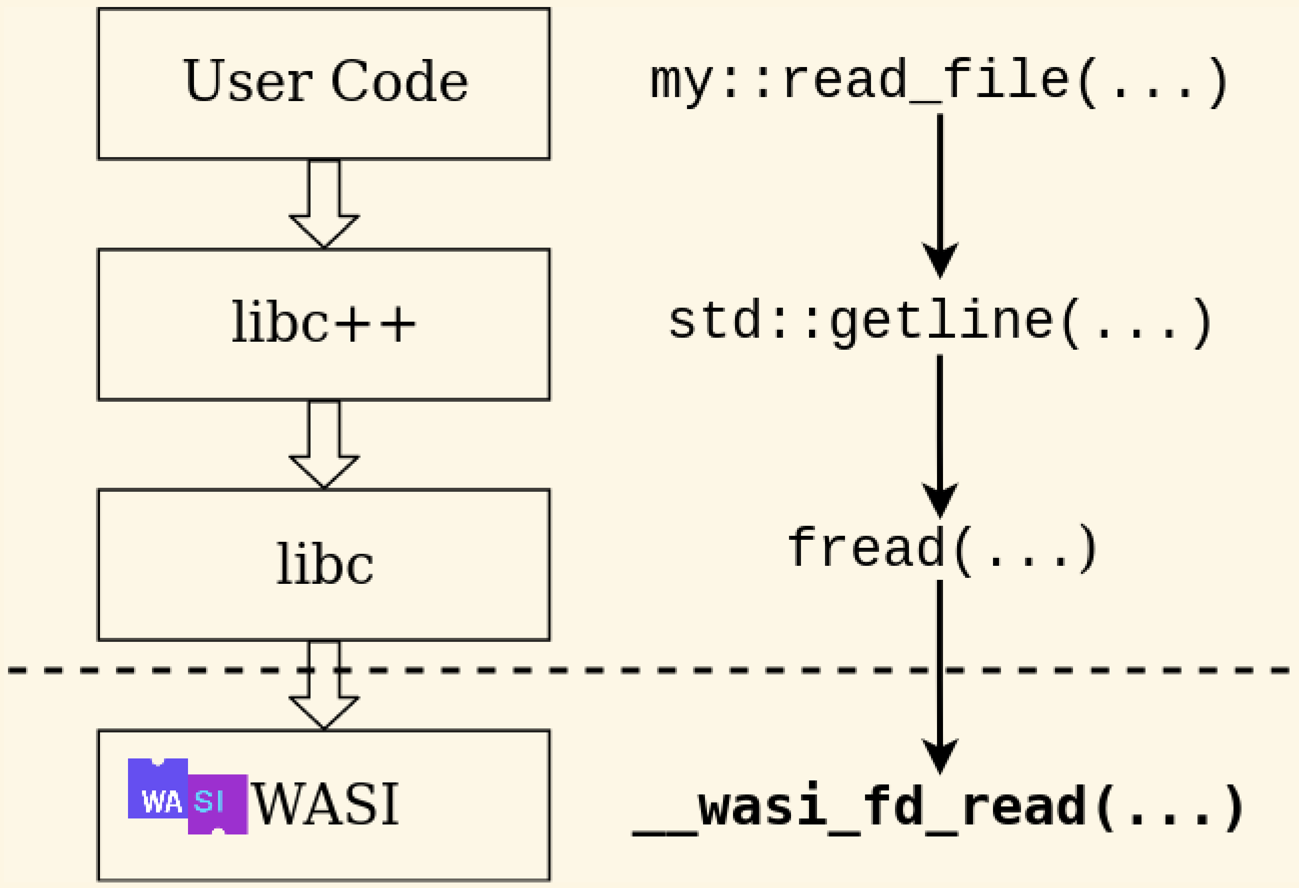
WebAssembly System Interface

WA SI

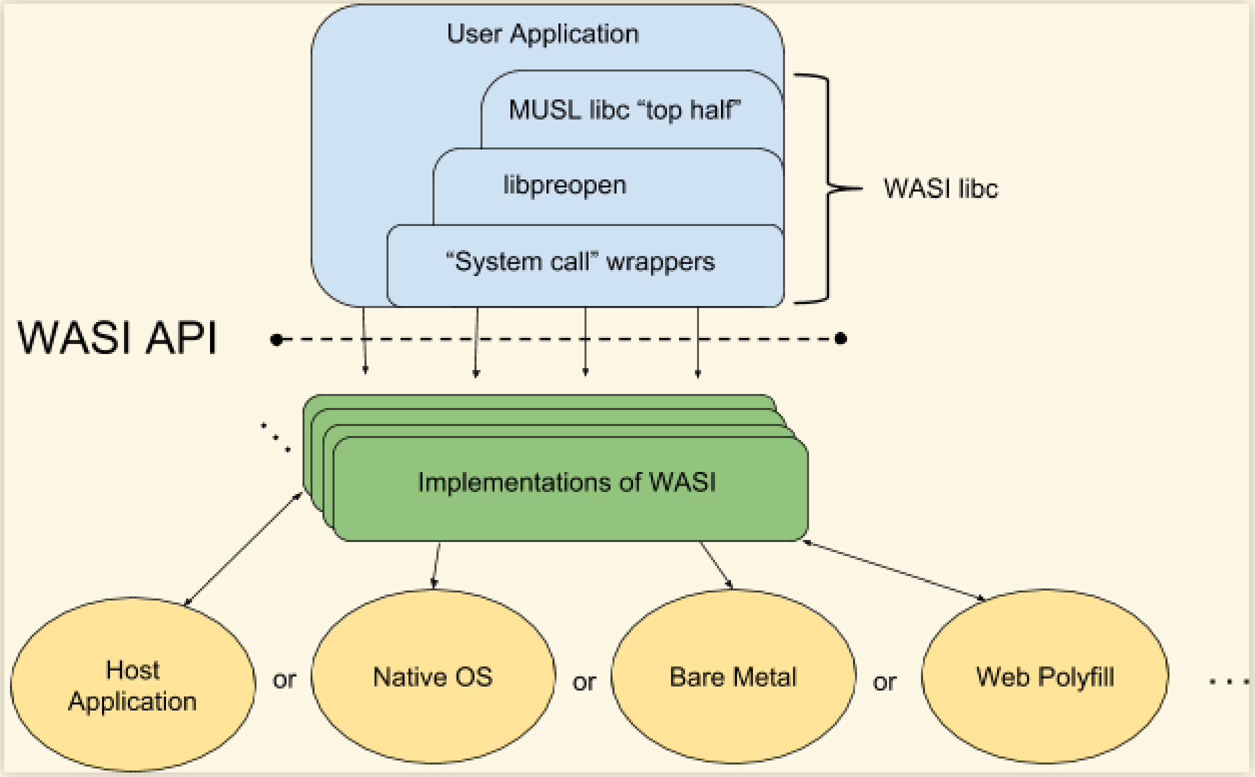
Reading a File



Reading a File



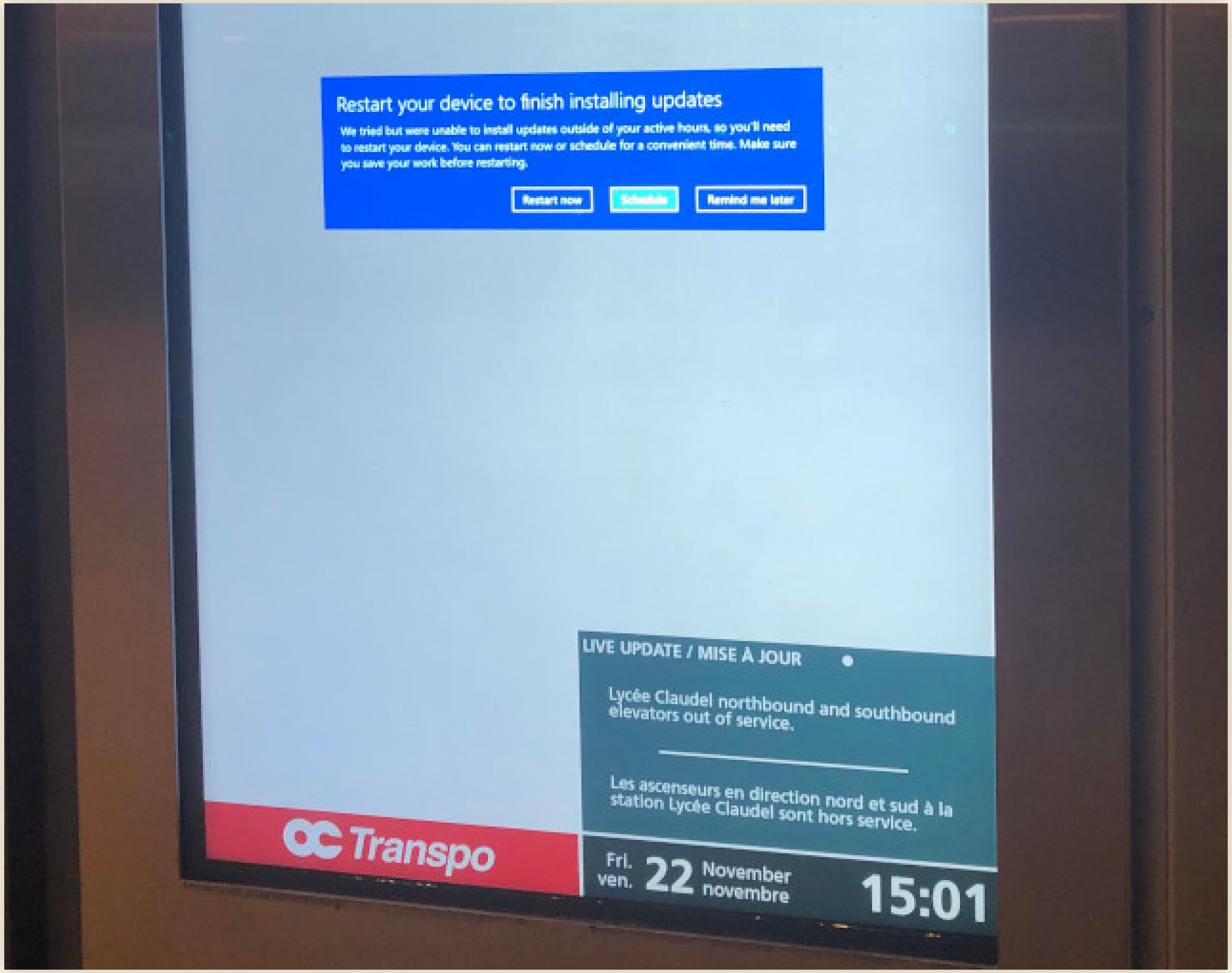
WASI Architecture



Current State

- WASI Core: work in progress
- Arguments, files/directories, time, sockets
- Experimental support in runtimes and toolchains

Live Demo!


Restart your device to finish installing updates

We tried but were unable to install updates outside of your active hours, so you'll need to restart your device. You can restart now or schedule for a convenient time. Make sure you save your work before restarting.

LIVE UPDATE / MISE À JOUR

Lycée Claudel northbound and southbound elevators out of service.

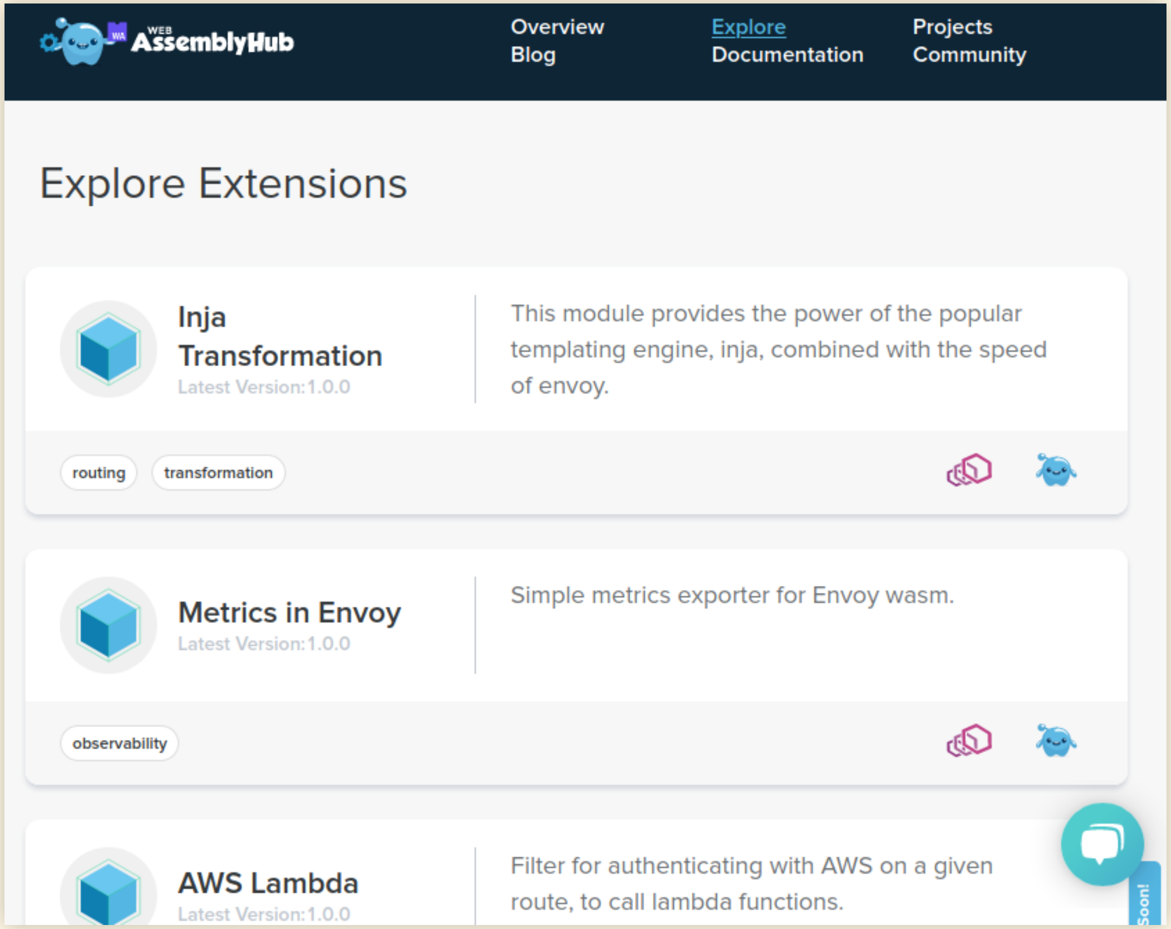
Les ascenseurs en direction nord et sud à la station Lycée Claudel sont hors service.

Present and Future of WebAssembly (outside the browser)

Buzzword Bingo Incoming

Plugins

Envoy proxy filters: C++, Lua and now WASM.



Explore Extensions

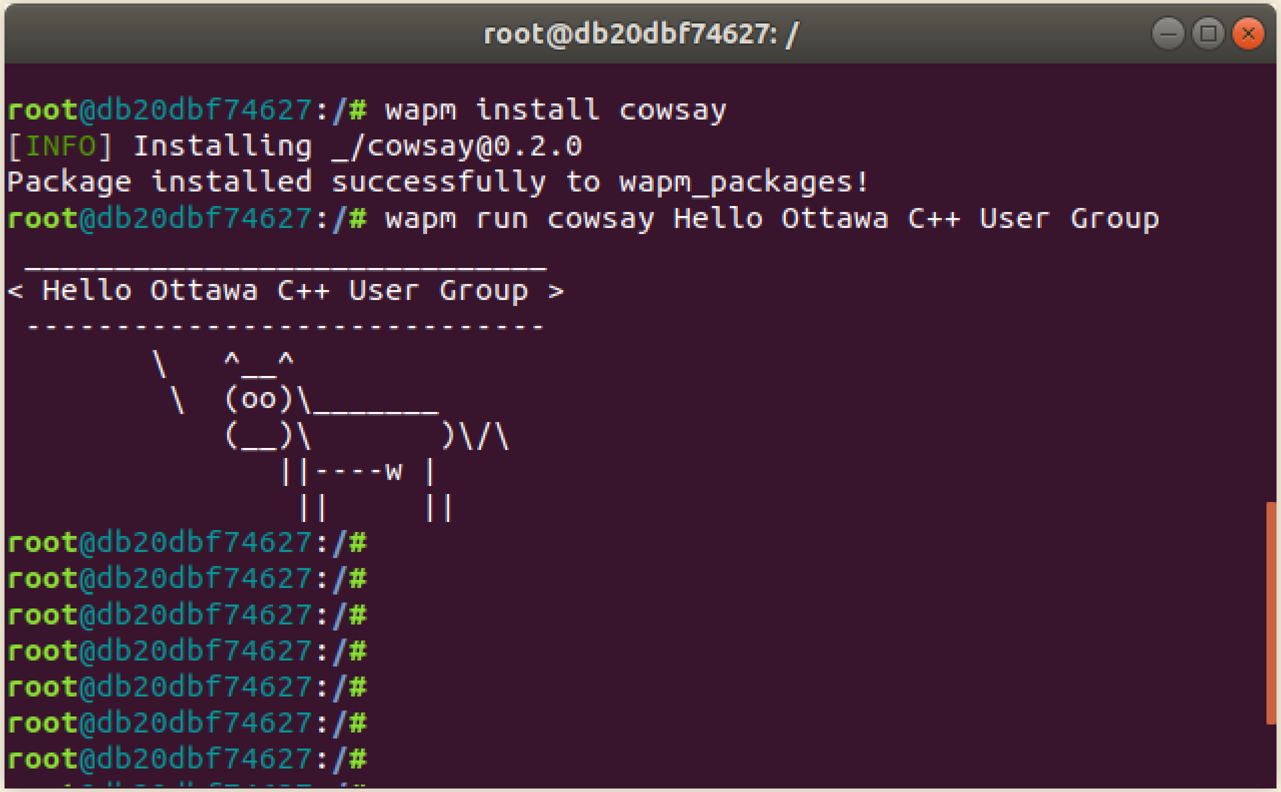
Inja Transformation
Latest Version:1.0.0
This module provides the power of the popular templating engine, inja, combined with the speed of envoy.
routing transformation

Metrics in Envoy
Latest Version:1.0.0
Simple metrics exporter for Envoy wasm.
observability

AWS Lambda
Latest Version:1.0.0
Filter for authenticating with AWS on a given route, to call lambda functions.

Command Line Applications

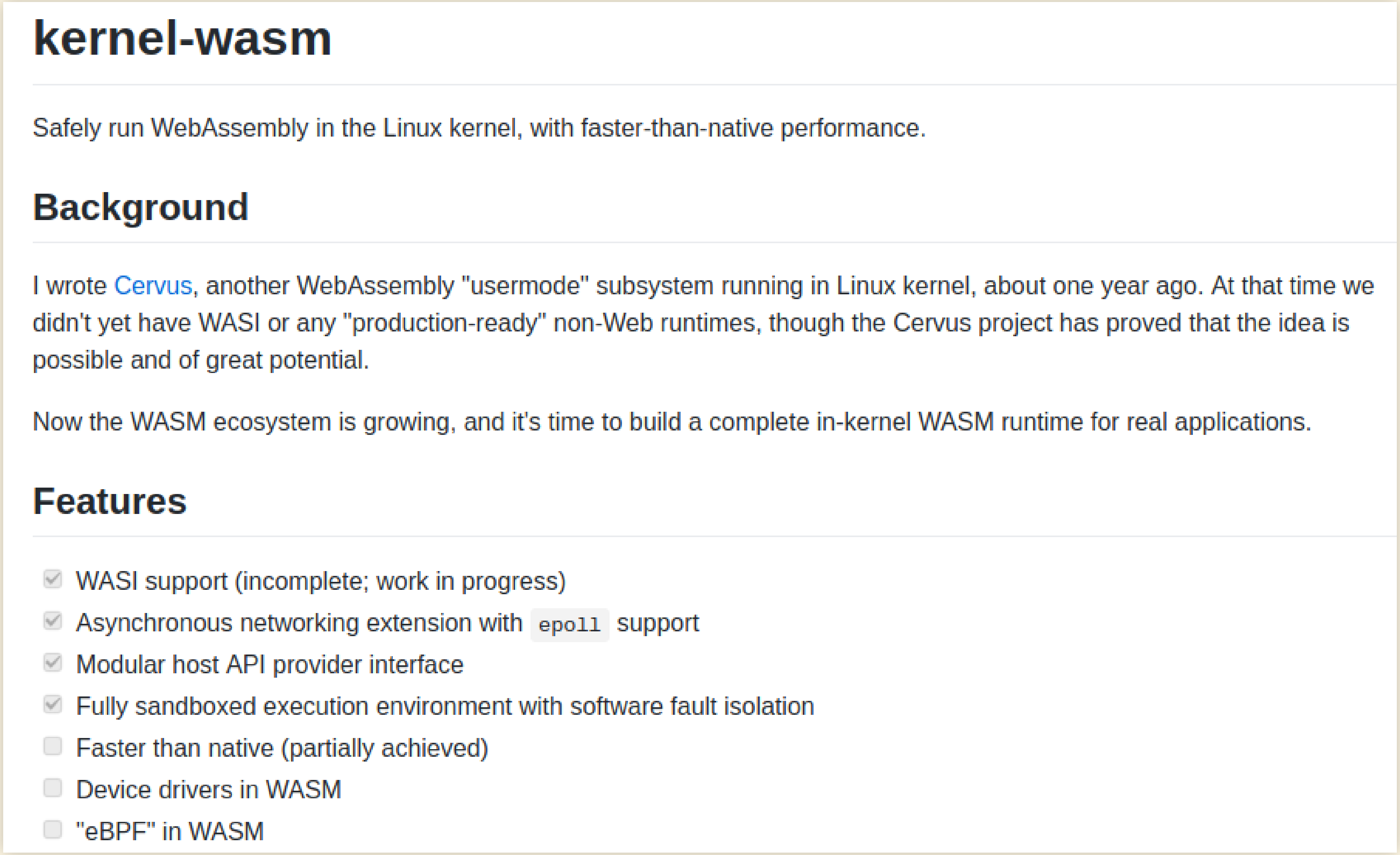
Wasmer + WebAssembly Package Manager (WAPM)



```
root@db20dbf74627:/# wapm install cowsay
[INFO] Installing _/cowsay@0.2.0
Package installed successfully to wapm_packages!
root@db20dbf74627:/# wapm run cowsay Hello Ottawa C++ User Group

 _____________________________
< Hello Ottawa C++ User Group >
 -----------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
root@db20dbf74627:/#
root@db20dbf74627:/#
root@db20dbf74627:/#
root@db20dbf74627:/#
root@db20dbf74627:/#
root@db20dbf74627:/#
root@db20dbf74627:/#
```

WASM in the Linux Kernel



kernel-wasm

Safely run WebAssembly in the Linux kernel, with faster-than-native performance.

Background

I wrote Cervus, another WebAssembly "usermode" subsystem running in Linux kernel, about one year ago. At that time we didn't yet have WASI or any "production-ready" non-Web runtimes, though the Cervus project has proved that the idea is possible and of great potential.

Now the WASM ecosystem is growing, and it's time to build a complete in-kernel WASM runtime for real applications.

Features

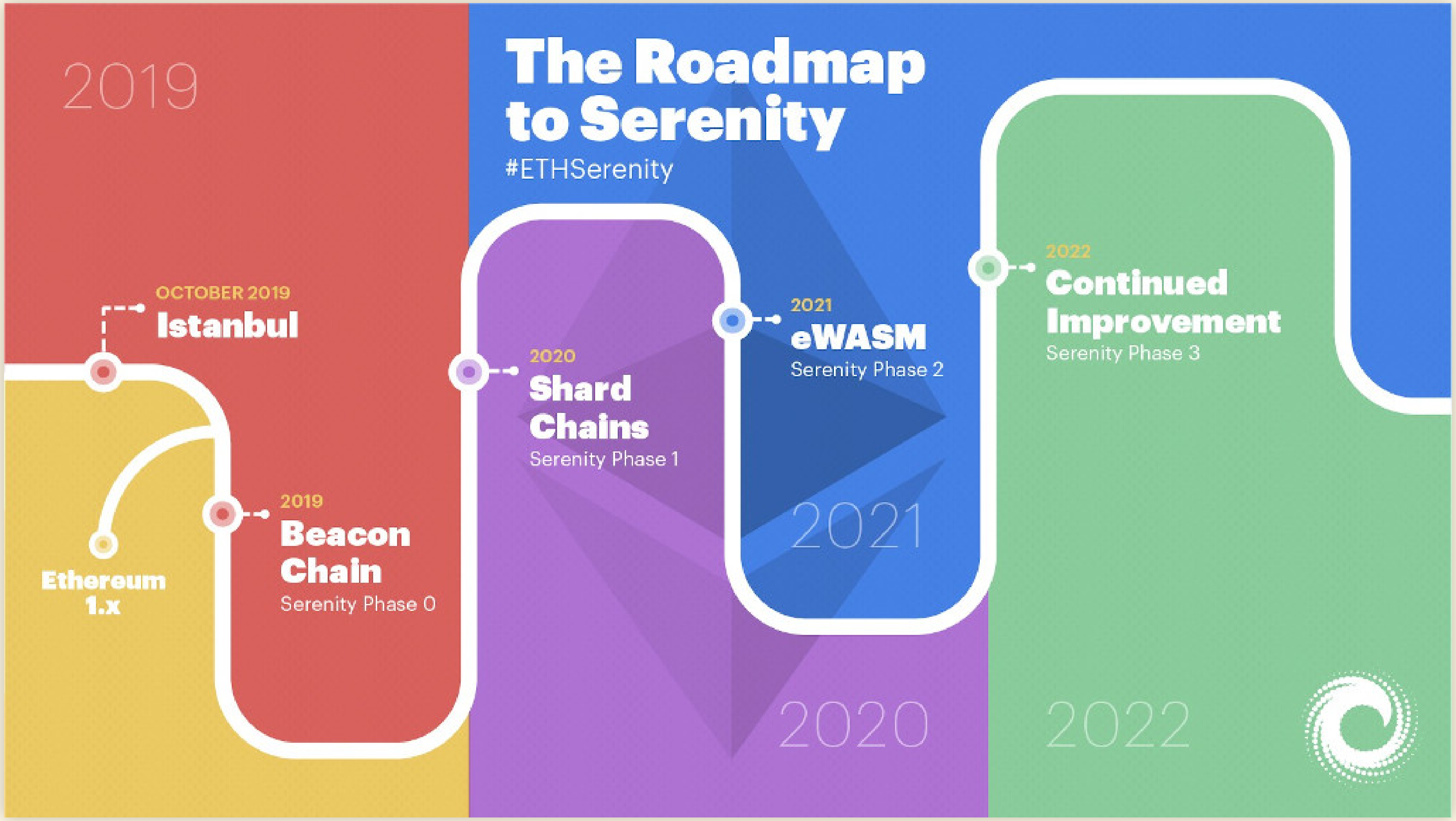
- [x] WASI support (incomplete; work in progress)
- [x] Asynchronous networking extension with `epoll` support
- [x] Modular host API provider interface
- [x] Fully sandboxed execution environment with software fault isolation
- [ ] Faster than native (partially achieved)
- [ ] Device drivers in WASM
- [ ] "eBPF" in WASM

IoT

WAMR + Application Framework

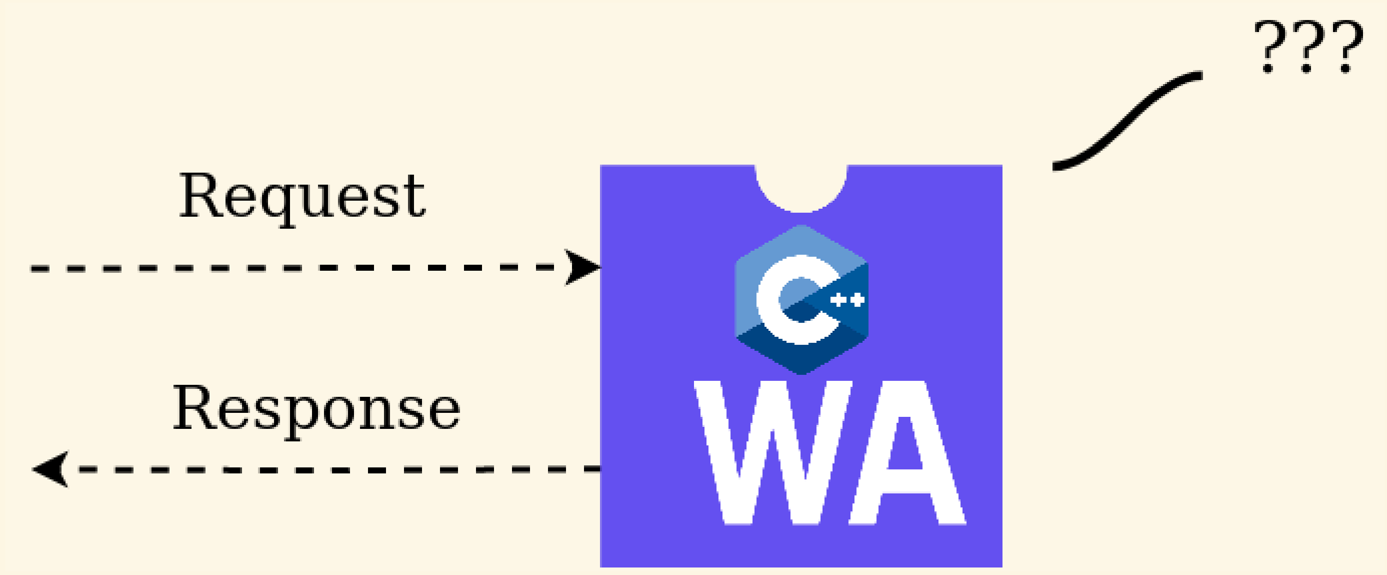
Blockchain

Ethereum WebAssembly (eWASM)

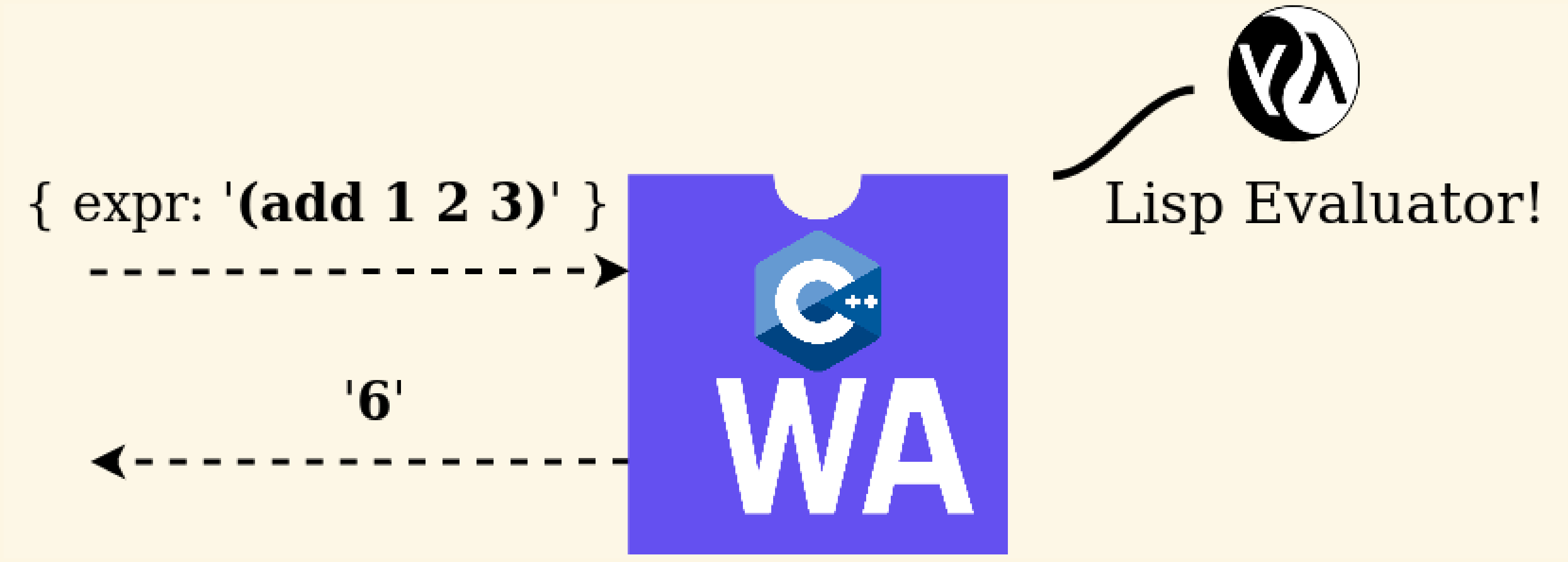


Live Demo!

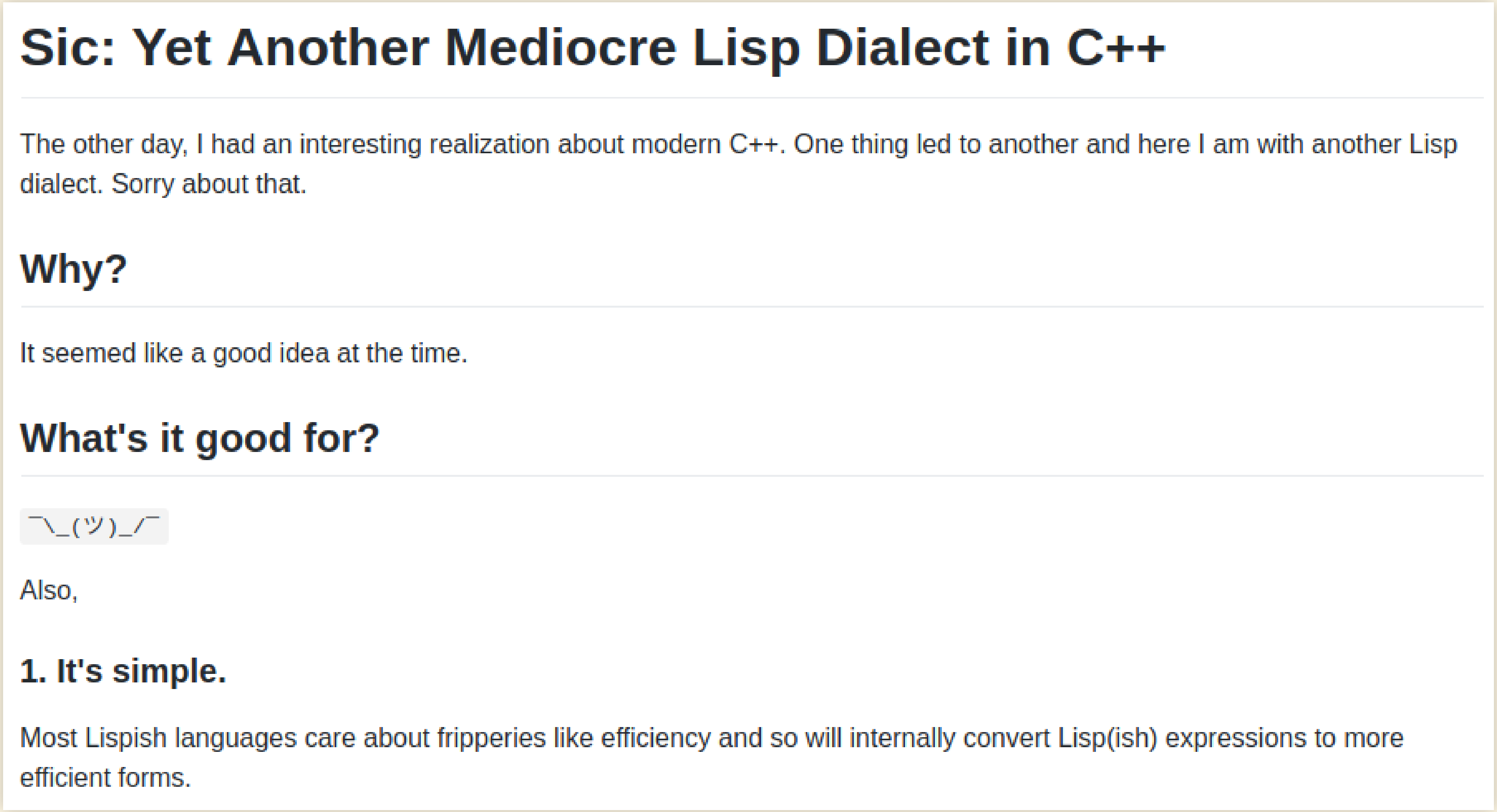
Web Service



Lisp as a Service



github.com/suetanvil/sic



Sic: Yet Another Mediocre Lisp Dialect in C++

The other day, I had an interesting realization about modern C++. One thing led to another and here I am with another Lisp dialect. Sorry about that.

Why?

It seemed like a good idea at the time.

What's it good for?

`¯\_(ツ)_/¯`

Also,

1. It's simple.

Most Lispish languages care about fripperies like efficiency and so will internally convert Lisp(ish) expressions to more efficient forms.

Getting Rid of Exceptions

- `throw ...` → `std::abort()`
- Comment out `try / catch` keywords

Dynamic Initialization

```
const Foo* foo = foo();
const Bar* bar = bar();
```

↓

```
Foo* foo = nullptr;
Bar* bar = nullptr;

void init() {
    foo = foo();
    bar = bar();
}
```

There Is No Escape From printf()

References

- webassembly.org
- github.com/webassembly
- wasi.dev
- webassembly.studio
- github.com/mbasso/awesome-wasm