

# WebAssembly is Cool!

(finally)



[D3wasm](#) - An experimental port of id Tech 4 engine to [Emscripten](#) / [WebAssembly](#)

Online demonstration running **Doom 3 Demo**

Hint: use HOME key instead of ESC key (go to main menu), and INSERT key instead of ` key (open console)

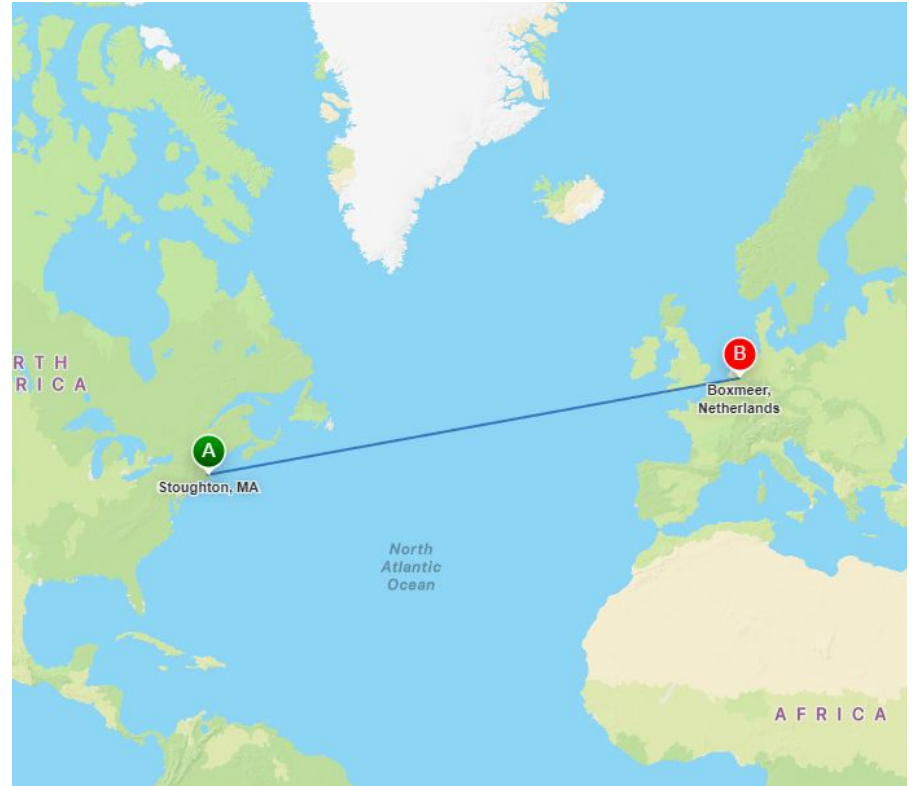
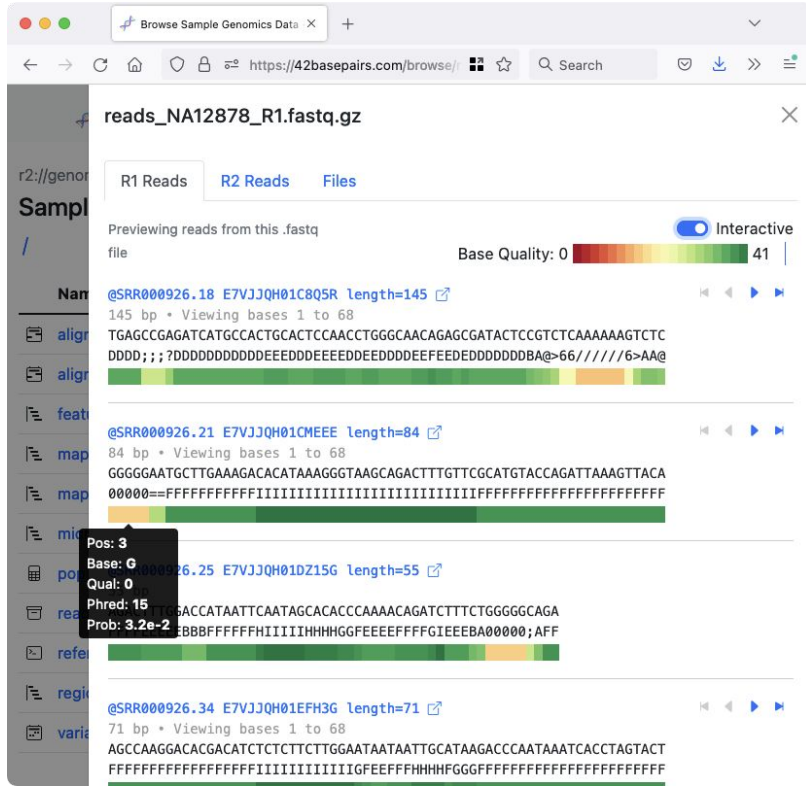
■ Show help



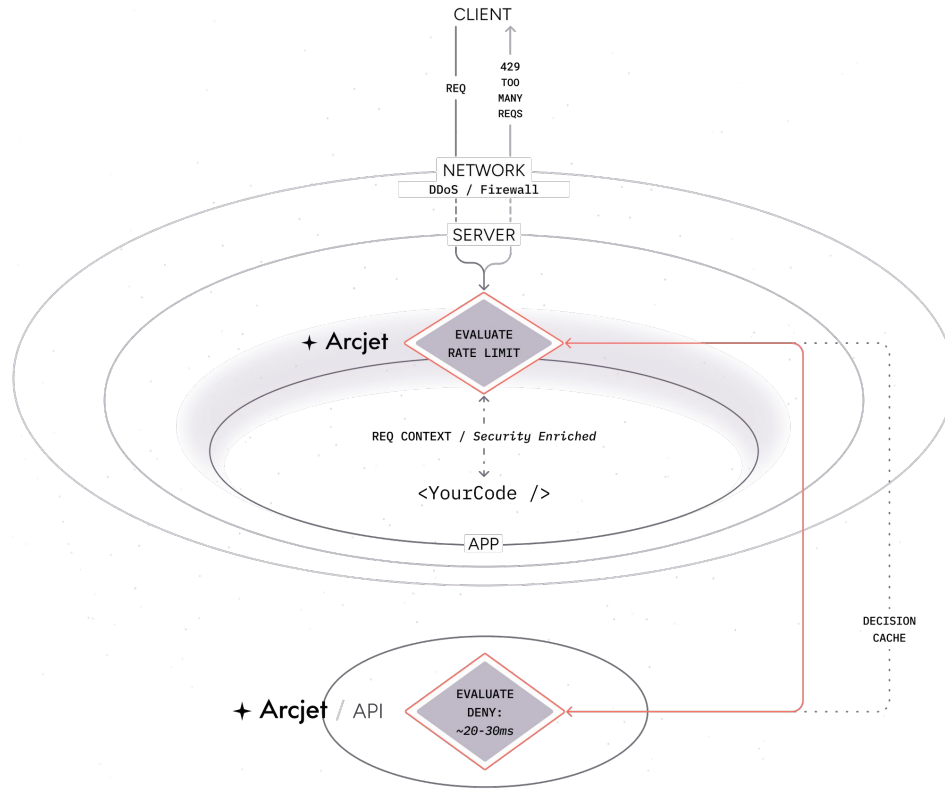
All information about this port, including purpose, source code, technical details, and legal info can be found on the [project](#) page.

# Doom at 60fps in your browser

# Scientific compute with zero install



# Security at near-native speeds





Internet  
Explorer



Sheep.exe



Recycle Bin



Network  
Neighborhood



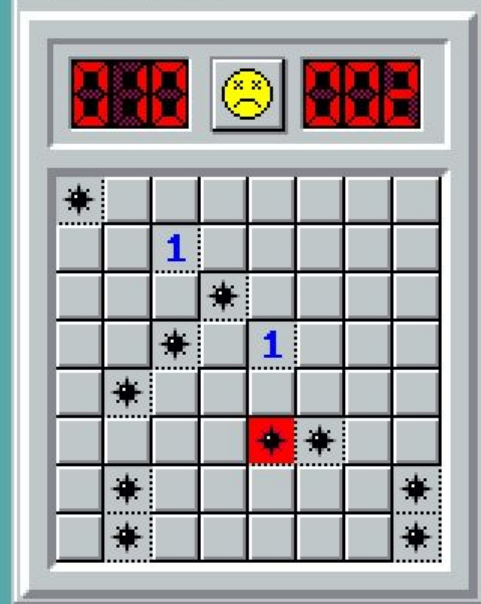
Online  
Services



The Microsoft  
Sound.wav



networking.bat



So who ***IS*** this guy?

(fair question)

# Jakob Heuser

Former Pinterest, LinkedIn

Co-founder, builder, maker

Wasm Enthusiast (obviously)



# Setting expectations & ground rules

— — —

Pro-Wasm doesn't mean feeding the hype cycle

See Wasm through a practical lens

Avoid a messy Q&A, let's talk in small groups after!



# Today

---

~~Guy plays DOOM 3 and Minesweeper~~

~~About that Jakob guy~~

What WebAssembly ISN'T / What WebAssembly IS

Practical use cases

The future & more inspiring stuff

# Keep Learning

— — —

Google I/O - WebAssembly: A new development paradigm for the web (2023)

<https://www.youtube.com/watch?v=RcHER-3gFXI>

NDC - The WebAssembly Component Model (2024)

[https://www.youtube.com/watch?v=\\_fKPvnhX-vI](https://www.youtube.com/watch?v=_fKPvnhX-vI)

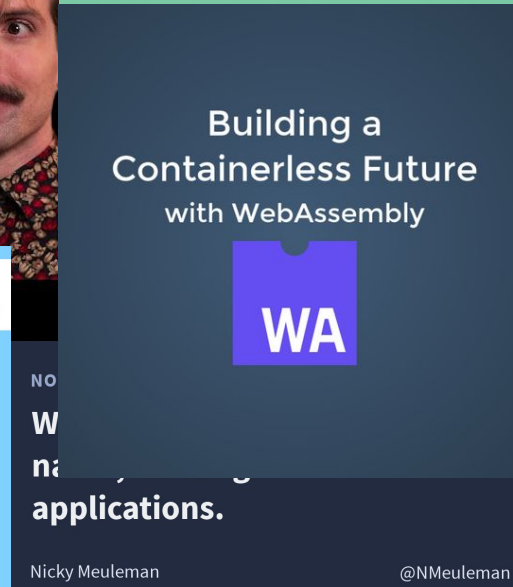
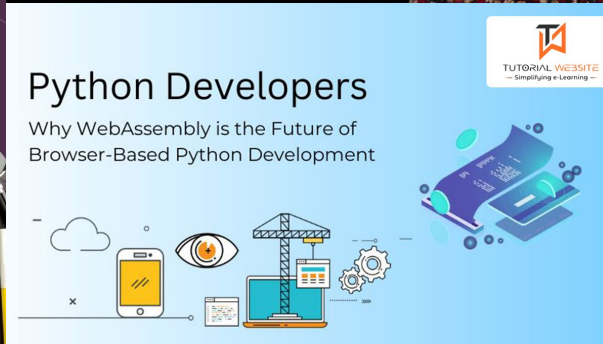
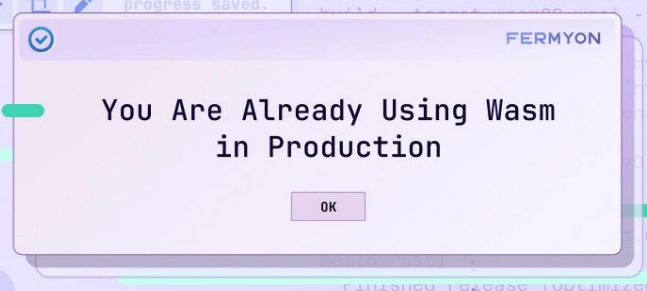
Devoxx UK - WebAssembly outside the browser (2024)

<https://www.youtube.com/watch?v=We1JKjjTFXI>



So what ***ISN'T*** WebAssembly?

(thanks for reading subtitles!)



# ***NOT*** Java in the browser 2.0

---

Secure by default (no, really)

Does not include a runtime

No direct system calls

**“If Wasm+WASI existed in 2008, we  
wouldn't have needed to create Docker.  
That's how important it is.”**

Solomon Hykes (co-founder of Docker)

# ***NOT*** replacing Docker

---

WebAssembly excels at single tasks

Docker excels at gnarly imperfect software running together

You're not swapping Docker with Wasm

***NOT*** replacing JavaScript (sorry)

---

Something must talk to the DOM

Most companies aren't going to add "and a binary" to JS builds



# So what *IS* WebAssembly?

(besides not being for just the web and also not being assembly)

# World's fastest Wasm history lesson

---

Announced 2015 / Launched 2017 / W3C 2019

Successor to technologies like asm.js & Emscripten

Now (Sept 17) on version 3.0 of the specification

# Cool, but what ***IS*** it?

---

A virtual instruction set architecture (virtual ISA)

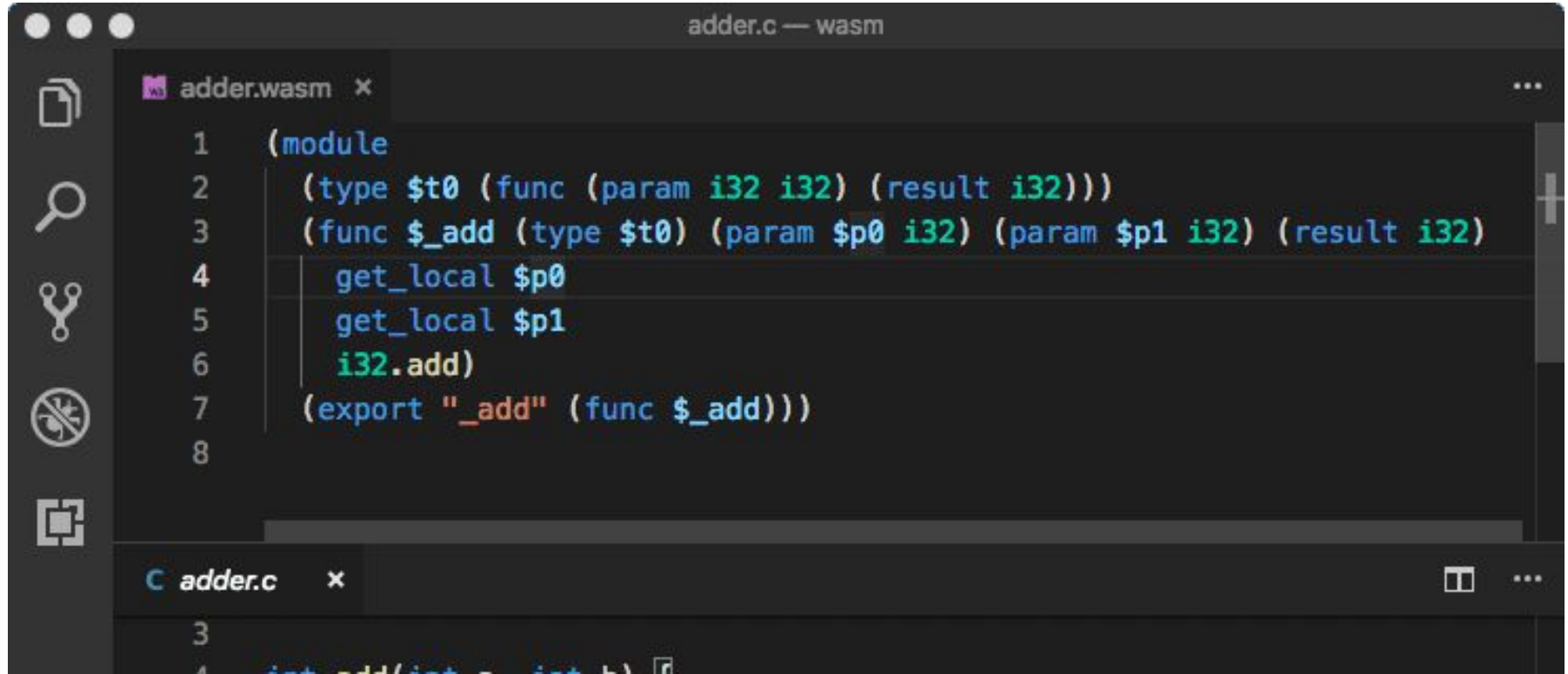
Uses Linear Memory with few "types": i32, i64, f32, f64...

Embeddable in a Host Environment

So let's go  
*deeper* into  
WebAssembly

# A virtual ISA with a stack-based design

---



The screenshot shows a code editor with two tabs: 'adder.wasm' and 'adder.c'. The 'adder.wasm' tab is active, displaying the following WebAssembly code:

```
1 (module
2   (type $t0 (func (param i32 i32) (result i32)))
3   (func $_add (type $t0) (param $p0 i32) (param $p1 i32) (result i32)
4     get_local $p0
5     get_local $p1
6     i32.add)
7   (export "_add" (func $_add)))
8
```

The 'adder.c' tab is partially visible at the bottom, showing the following C code:

```
3
4 int add(int a, int b) {
```

# That compiles to a binary instruction format

---

## TEXTUAL FORMAT

```
(module
  (func $addTwo (param i32 i32)
    (result i32)
    (i32.add
      (get_local 0)
      (get_local 1)))
  (export "addTwo" $addTwo))
```

=

## BINARY FORMAT

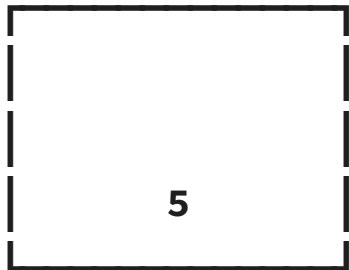
```
48 83 EC 08
8B CF
8B C1
03 C6
66 90
48 83 C4 08
C3
```

# The stack design

---

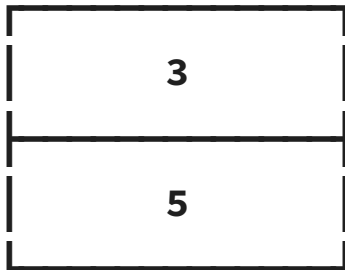
**PROGRAM:** (i32.const 5) (i32.const 3) (i32.add)

**Step 1:** i32.const 5



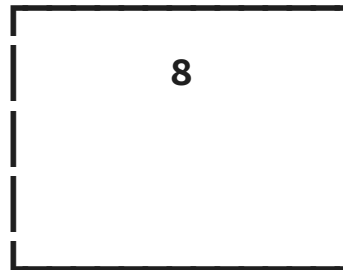
**Stack:** [5]

**Step 2:** i32.const 3



**Stack:** [5, 3]

**Step 3:** i32.add



**Stack:** [8]

DevTools - localhost:5000/mandelbrot

ElementsConsoleSourcesNetworkPerformanceMemoryApplicationSecurityLighthouse

Page >>mandelbrot.jsmandelbrot.wasm xmandelbrot.cc

toplocalhost:5000mandelbrotmandelbrot.jsmandelbrot.wasmfile://

0x010fc8  
0x010fc9  
0x010fcb  
0x010fcd  
0x010fcf  
0x010fd1  
0x010fd2  
0x010fd4  
0x010fd6  
0x010fd9  
0x010fdd  
0x010fde  
0x010fe0  
0x010fe1  
0x010fe5  
0x010fe7  
0x010fea  
0x010feb  
0x010fed  
0x010fee  
0x010fef  
0x010ff1

(func \$SDL\_SetRenderDrawColor (;444;) (param \$var0 i32) (param \$var1 i32) (param \$var2 i32) (param \$var3 i32))  
    block \$label1  
        block \$label0  
            local.get \$var0  
            i32.eqz  
            br\_if \$label0  
            local.get \$var0  
            i32.load  
            i32.const 64641  
            i32.eq  
            br\_if \$label1  
        end \$label0  
        i32.const 8833  
        i32.const 0  
        call \$SDL\_SetError  
        drop  
        i32.const -1  
        return  
    end \$label1  
    local.get \$var0

Bytecode position 0x10fd1Coverage: n/a

Pause on caught exceptions

Debugger paused

Watch

Breakpoints

mandelbrot.cc:31  
std::complex<double> point((double)x ...

Scope

Module

env.memory: Uint8Array(16777216) [101, ...

globals: {global0: 5306976, global1: 65...

instance: Instance {}

Local

var0: 5314352

var1: 111

var2: 149

var3: 224

var4: 255

ConsoleSearchProtocol monitorWhat's NewMemory Inspector x

mandelbrot.wasm x

<0x00511730>

005116E001 00 00 00 01 00 00 00 00 00 00 . . . . .

005116EC00 00 00 00 B2 99 00 00 00 00 00 . . . . .

005116F804 18 16 16 80 07 69 00 00 00 00 . 0 0 0 . . i . . . .

0051170400 00 F0 3F 00 00 00 00 00 00 F0 3F . . . . .

Little Endian

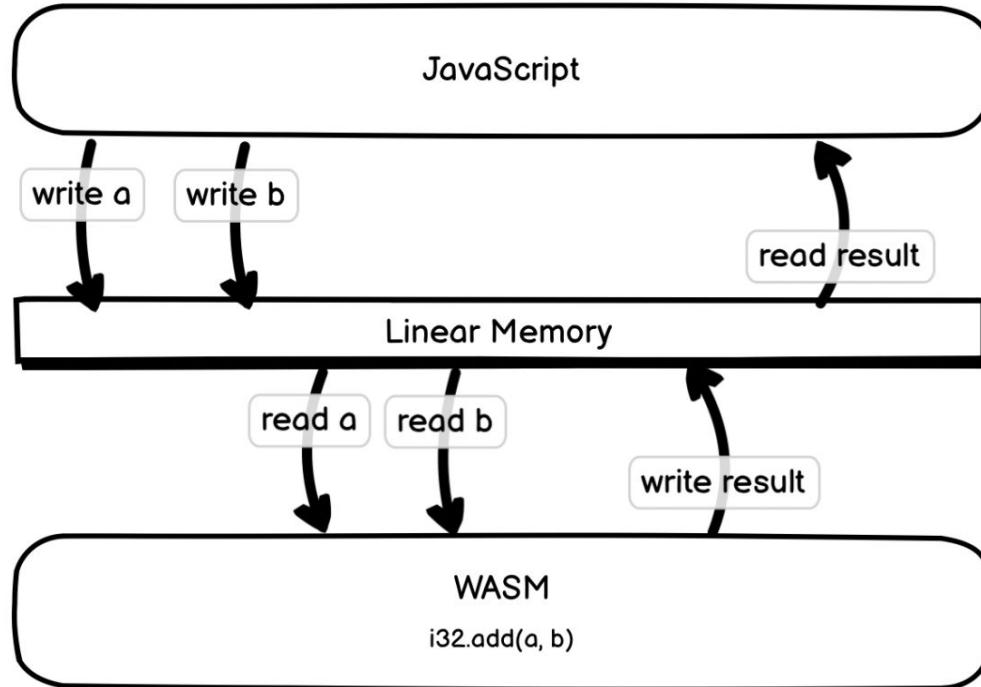
Integer 8-bitdec129 / -127

Float 32-bitdec0.00



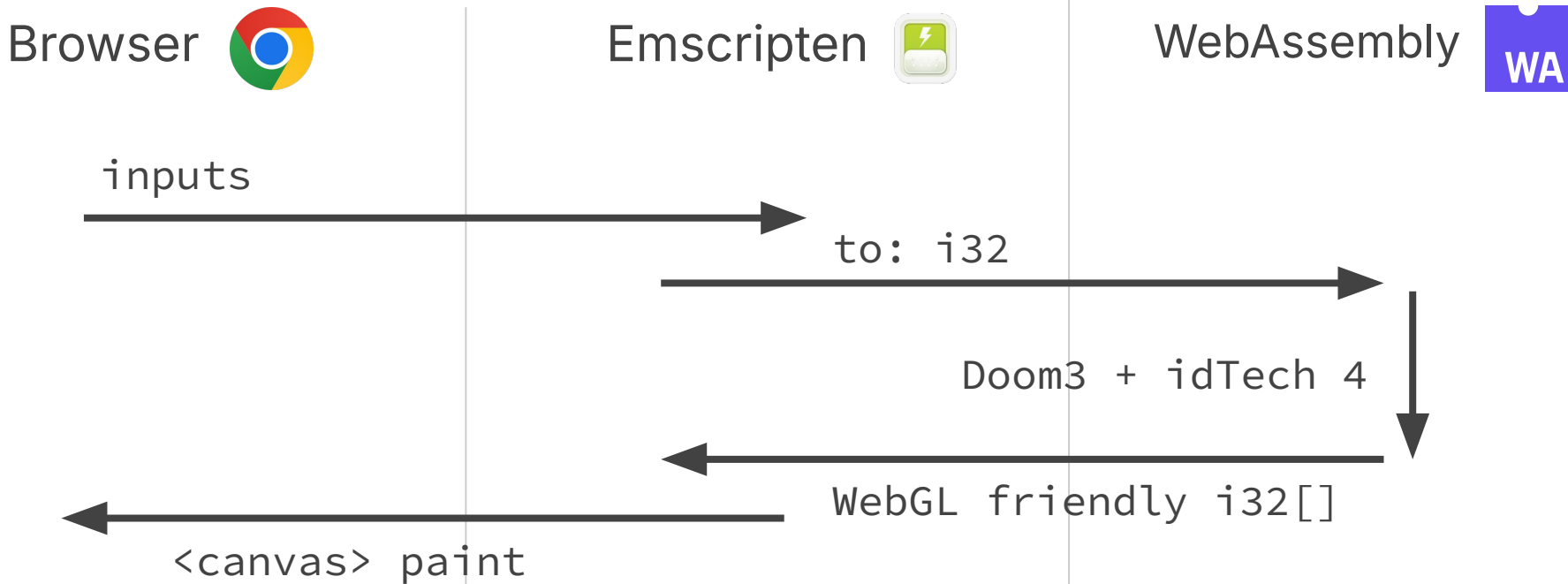
# Utilizing Linear Memory

---

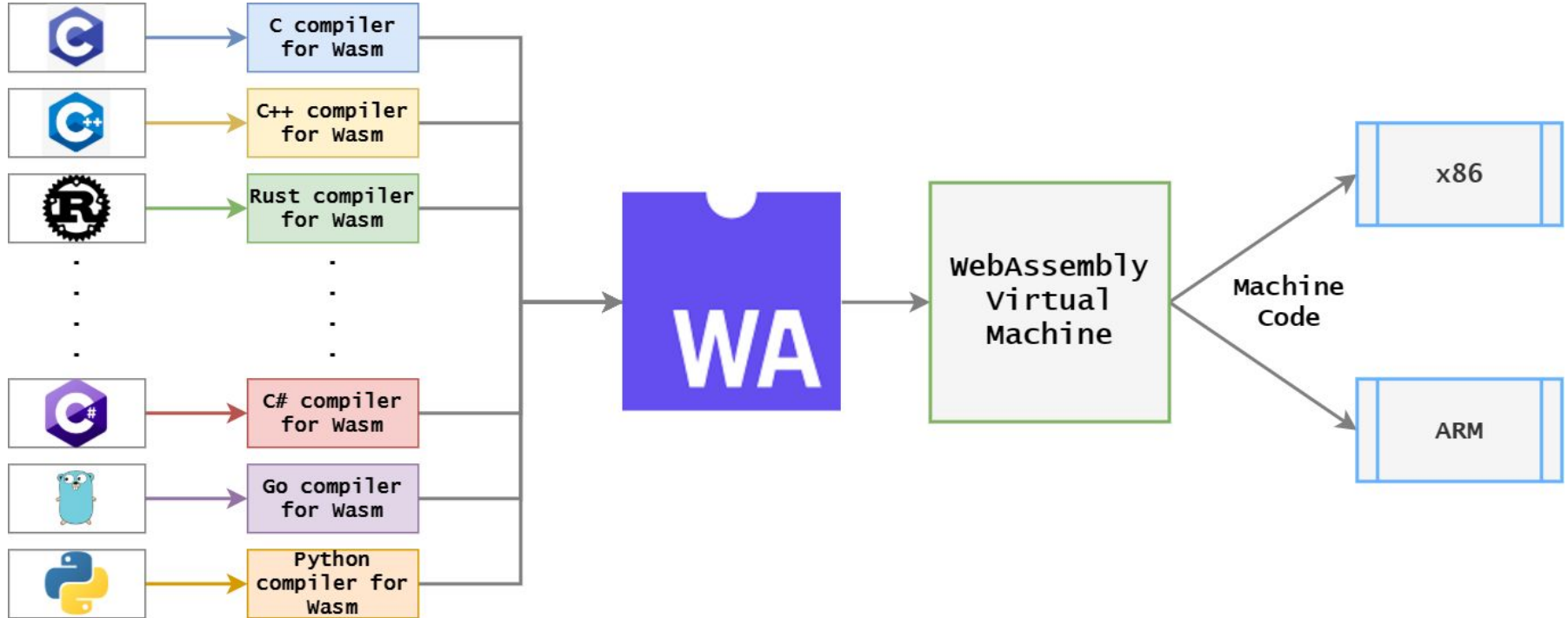


# Doom3 is linear memory & WebGL

---

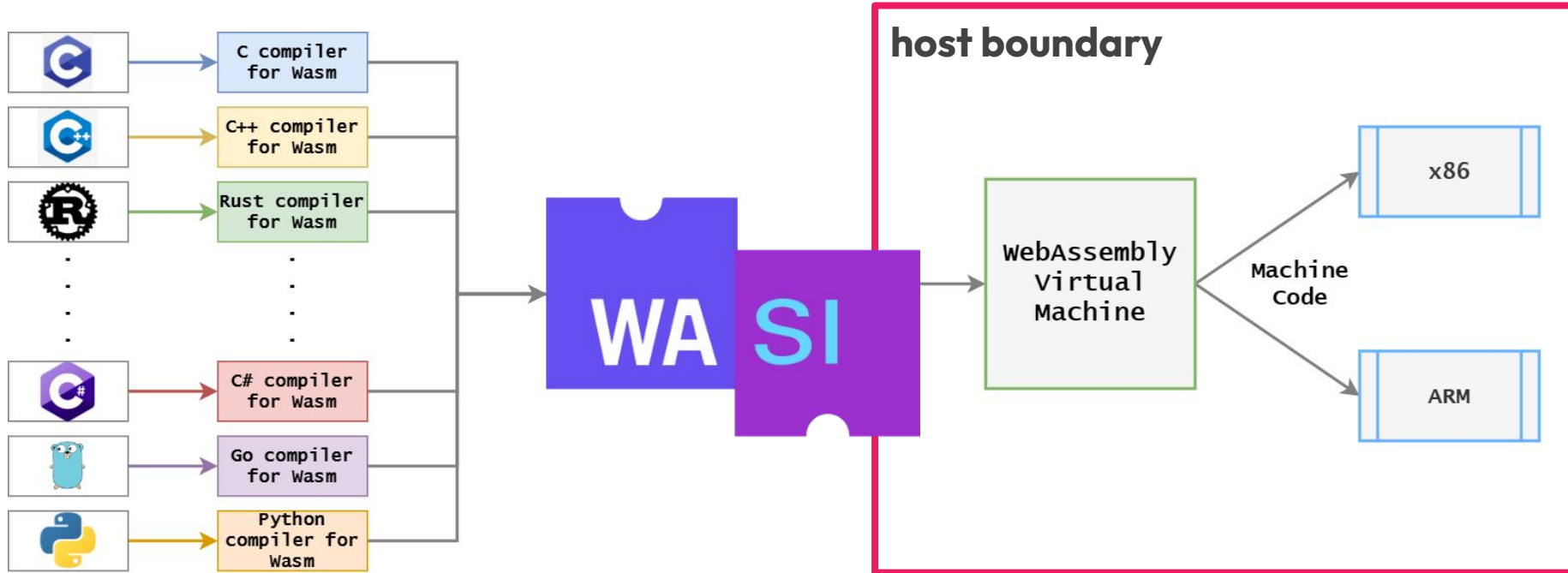


# Gains superpowers through WASI



# Embedded in a Host Environment

---



# Without compromising security

---

Traps at the Wasm level  $\Rightarrow$  Exceptions in host

Module isolation puts every Wasm in its own memory

Attack surface area defined by features allowed

# WebAssembly at its Best

(the kinds of problems Wasm was meant for)

\_\_\_\_\_

Pos: 3  
Base: G  
Qual: 0  
Phred: 15  
Prob: 3.2e-2

```
min_len: 9; max_len: 19; avg_len: 14.00; 1 distinct
POS      #bases  %A      %G      %T      %N
ALL       28      32.1     10.7     32.1     25.0     0.0
1         2       50.0     0.0      0.0     50.0     0.0
2         2       0.0      0.0      0.0    100.0     0.0
3         2       50.0     0.0     50.0     0.0     0.0
4         2       0.0      0.0    100.0     0.0     0.0
5         2       50.0     50.0     0.0     0.0     0.0
6         2       0.0      0.0     50.0     50.0     0.0
7         2       50.0     50.0     0.0     0.0     0.0
8         2       50.0     0.0     0.0     50.0     0.0
9         2       50.0     0.0     50.0     0.0     0.0
```

# Vector search in edge computing: Voy

```
🔥 Welcome to voy
⚙️ Loading voy ...
⚙️ voy is loaded ✓ ...
⚙️ voy is indexing [ "That is a very happy Person", "That is a Happy Dog",
"Today is a sunny day" ] ...
⚙️ voy is indexed ✓ ...
⚙️ voy is searching for the nearest neighbor for "That is a happy person" ...
⚙️ voy similarity search result 📌 "That is a very happy Person"
🌟 Done
```



# Beyond the browser: universal runtimes

— — —

Call  code from your  apps.

The cross-language framework for building with WebAssembly

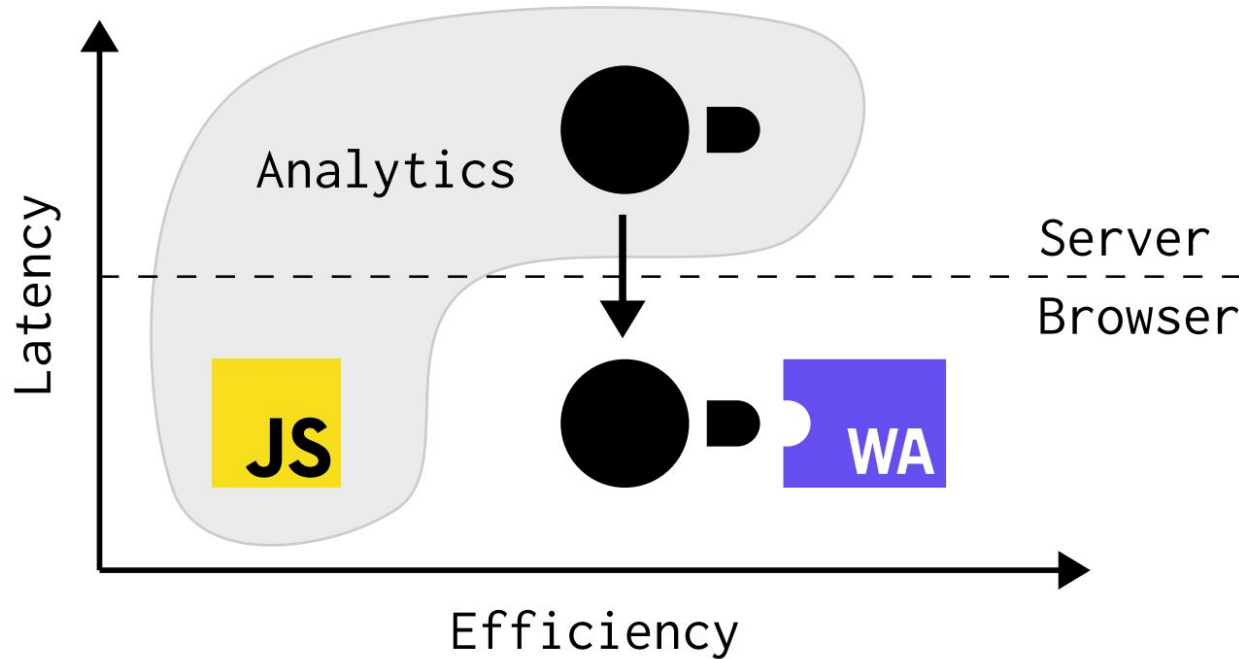
[Read the docs](#)

Quickly embed into officially supported languages:



# Private analytics anywhere: DuckDB-Wasm

---



# The Future of Wasm

(cool things to keep an eye on)

# Containerization

---

## **Wasmer & Boxer & wasmtime**

Universal Wasm Binary

A Host Environment with WASI

A "different" container



# Portable Compute

— — —

PostgreSQL UDFs with **Extism**

Universal build tooling with **Moonrepo**

# WASI 0.2: Upgraded host interaction (Jan 2024)

---



Proposal	Versions
<a href="https://github.com/WebAssembly/wasi-io">https://github.com/WebAssembly/wasi-io</a>	0.2.0
<a href="https://github.com/WebAssembly/wasi-clocks">https://github.com/WebAssembly/wasi-clocks</a>	0.2.0
<a href="https://github.com/WebAssembly/wasi-random">https://github.com/WebAssembly/wasi-random</a>	0.2.0
<a href="https://github.com/WebAssembly/wasi-filesystem">https://github.com/WebAssembly/wasi-filesystem</a>	0.2.0
<a href="https://github.com/WebAssembly/wasi-sockets">https://github.com/WebAssembly/wasi-sockets</a>	0.2.0
<a href="https://github.com/WebAssembly/wasi-cli">https://github.com/WebAssembly/wasi-cli</a>	0.2.0
<a href="https://github.com/WebAssembly/wasi-http">https://github.com/WebAssembly/wasi-http</a>	0.2.0

# Wasm 3.0: Hello features (Sept 2025)

---

64 bit address space

Native Garbage Collector

Fully Deterministic



# **WebAssembly Specification**

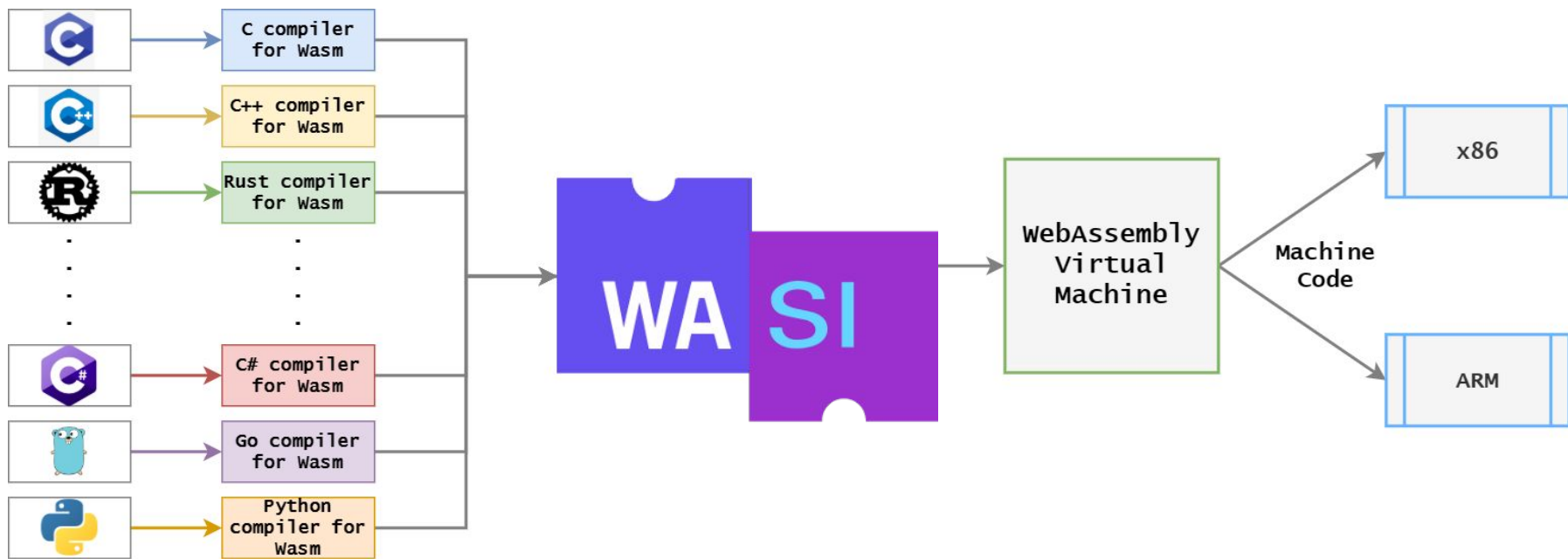
***Release 3.0 (2025-09-17)***

# Bringing It All Home

(and answering “when do I Wasm?”)



**You can  
Wasm today!**





**[jakob@codedrift.com](mailto:jakob@codedrift.com)**

Tell him what you liked about this!

# Appendix & Links

# Examples

— — —

[Doom3 Demo](#)

[42 Base Pairs data](#)

[Arcjet Example](#)

[Windows 98 Emulated](#)

# Examples (2)

---

[BioWasm](#)

[Voy](#)

[Extism](#)

[DuckDB](#) & [DuckDB-Wasm](#)

[Moonrepo](#)

# Examples (3)

— — —

[Wasmer](#)

[Boxer](#)

[Wasmtime](#)

# Foundational Technologies

— — —

[asm.js](#)

[Emscripten](#)



# Specifications

— — —

[WASI 0.2](#)

[Wasm 3.0](#)

# Even More Reading

---

[When is WebAssembly Going to Get DOM Support? \(hn\)](#)

[Wasm cut figma's load time by 3x \(figma\)](#)

[Shopify Functions using Wasm \(shopify\)](#)

[WASMs Linear Memory Model \(researchgate\)](#)

[Debugging WebAssembly \(chrome\)](#)

[Awesome Wasm Langs \(github\)](#)