BACKEND WEBASSEMBLY APPS

PETER LARSSON

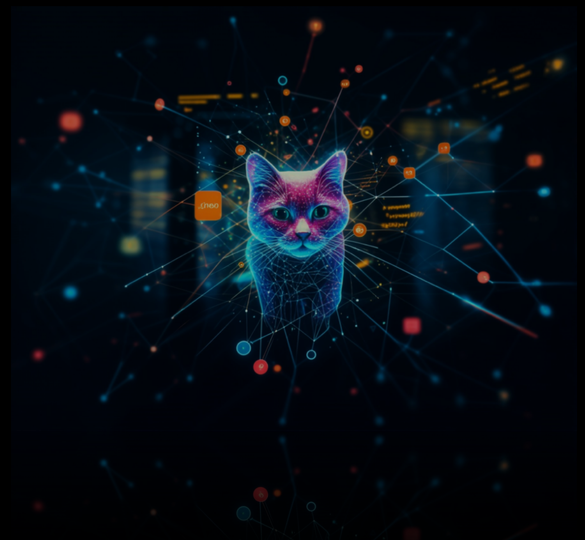
CADEC 2025.01.23 & 2025.01.29 | CALLISTAENTERPRISE.SE

CALLISTA

This page is intentionally left blank.

“The frontend space is always moving in every direction at the same time, this is known as Schrodinger’s frontend, depending on when you look at it and what intentions you have – you may think you’re looking at the backend.”

– bryanrasmussen, Hacker News



“WEBASSEMBLY IS NOT A WEB TECHNOLOGY”

– Dr. Andreas Rossberg, WASM Co-designer

BRIEF INTRODUCTION TO WEBASSEMBLY (WASM)

Wasm is a standardized byte code format and virtual instruction set architecture.

W3C WebAssembly WG + CG and System Interface Subgroup Charter

2015 Originated at Mozilla to complement JavaScript

2017 Supported by all main browsers

2019 W3C Recommendation, Wasm System Interface, WASI Preview 1 (Posix)

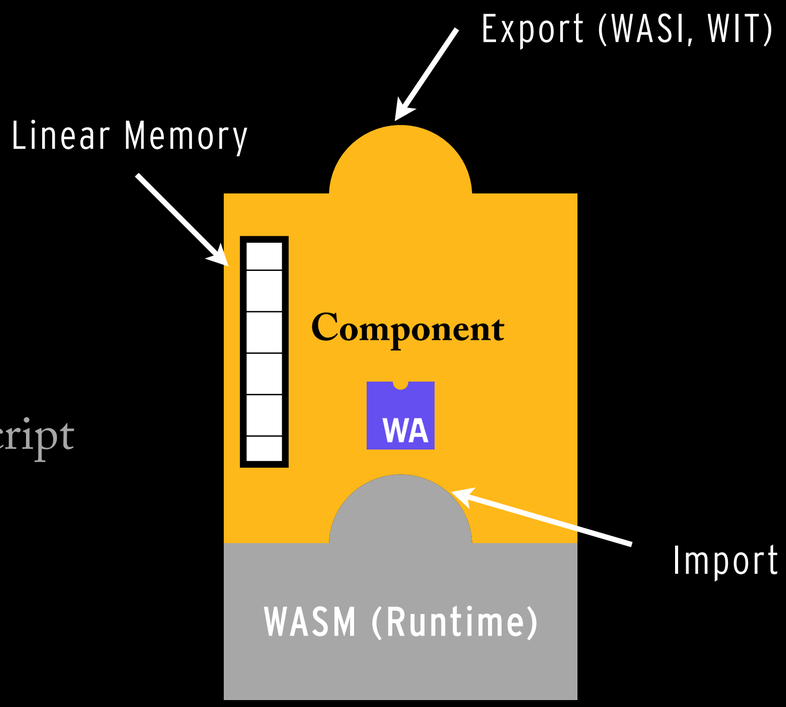
2024 WASI & Component Model Preview 2, Wasm Interface Type, WIT IDL

WASM, WASI DESIGN GOALS

Secure, Efficient, Portable and Modular

- Safe: predictable with validated code in a memory-safe sandboxed environment
- Polyglot: agnostic to language and programming model
- Fast: lightweight with near native code performance

C/C++
Kotlin
Python
Scala
Rust
Java
Typescript
JavaScript
Go
Grain
.NET
Swift
Ruby
MoonBit



Intel
MIPS
PowerPC
ARM
RISC-V

World Definition (WASI, WIT)

```
package component:example@0.1.0;

world example {
    import wasi:cli/stdout@0.2.3;
    import wasi:cli/environment@0.2.3;
    export wasi:cli/run@0.2.3;
    include wasi:http/proxy@0.2.3;
}
```

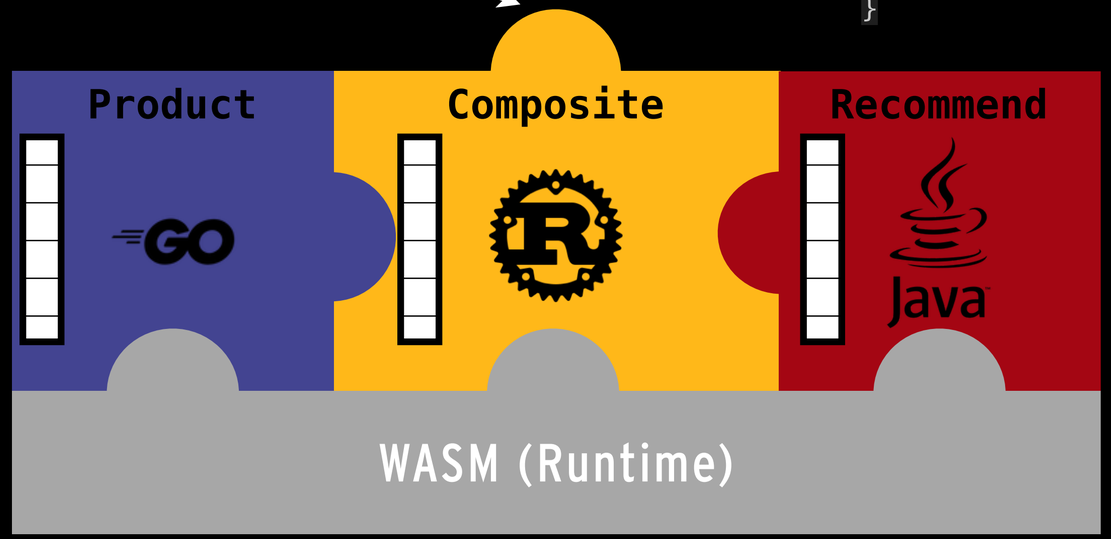
COMPONENT MODEL & COMPOSABLE WIT WORLDS

```
package product:composite@0.1.0;

world composite {
    import product:service/api@0.1.0;
    import recommendation:service/api@0.1.0;
    include wasi:http/proxy@0.2.3;
}
```

COMPONENT MODEL & COMPOSABLE WIT WORLDS

```
package product:composite@0.1.0;

world composite {
    import product:service/api@0.1.0;
    import recommendation:service/api@0.1.0;
    include wasi:http/proxy@0.2.3;
}
```

```
package recommendation:service@0.1.0;

interface api {
    record recommendation {
        author: string,
        rate: u32,
        content: string
    }
    get: func(product-id: u32) -> list<recommendation>;
}

world recommendation {
    export api;
}
```

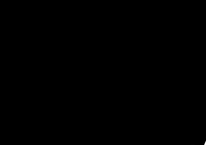



```
package product:service@0.1.0;

interface api {
    record product {
        id: u32,
        name: string,
        weight: u32
    }
    get: func(id: u32) -> option<product>;
}

world product {
    export api;
}
```

USE CASES

- ✓ Structured Monoliths
- ✓ Microservices
- ✓ Serverless Functions
- ✓ AI (LLM), Edge, Blockchain and IoT Applications
- ✓ Dynamic Extensions & Plugins

lightweight containerization



fast safe containers

CLOUD NATIVE
COMPUTING FOUNDATION

safe server side client code

determinism for consensus

portable containerization

AI controller Interface

portable universal platform

legacy unsafe code

client-side sandboxing

AI inference on the edge

CALLISTA

WHY CARE

HACKER ATTACKS AND BUGS

HACKER ATTACKS AND BUGS

Dependency confusion attacks (2021-ongoing)

Crowdstrike (2024)

Log4Shell (December 2021)

Cl0p MOVEit transfer attacks (May-June 2023)

XZ Utils backdoor (2024)

Kaseya VSA ransomware attack (July 2021)

Codecov backdoor (April 2021)

Okta support system breach (October 2023)

NPM package typosquatting attacks (ongoing)

3CX Desktop App (March 2023)

PyPI package attacks (ongoing)

HACKER ATTACKS AND BUGS

Dependency confusion attacks (2021-ongoing)

Crowdstrike (2024)

Log4Shell (December 2021)

Cl0p MOVEit transfer attacks (May-June 2023)

XZ Utils backdoor (2024)

Kaseya VSA ransomware attack (July 2021)

Codecov backdoor (April 2021)

Okta support system breach (October 2023)

NPM package typosquatting attacks (ongoing)

3CX Desktop App (March 2023)

PyPI package attacks (ongoing)

HACKER ATTACKS AND BUGS

Dependency confusion attacks (2021-ongoing)

Crowdstrike (2024)

Log4Shell (December 2021)

Cl0p MOVEit transfer attacks (May-June 2023)

XZ Utils backdoor (2024)

Kaseya VSA ransomware attack (July 2021)

Codecov backdoor (April 2021)

Okta support system breach (October 2023)

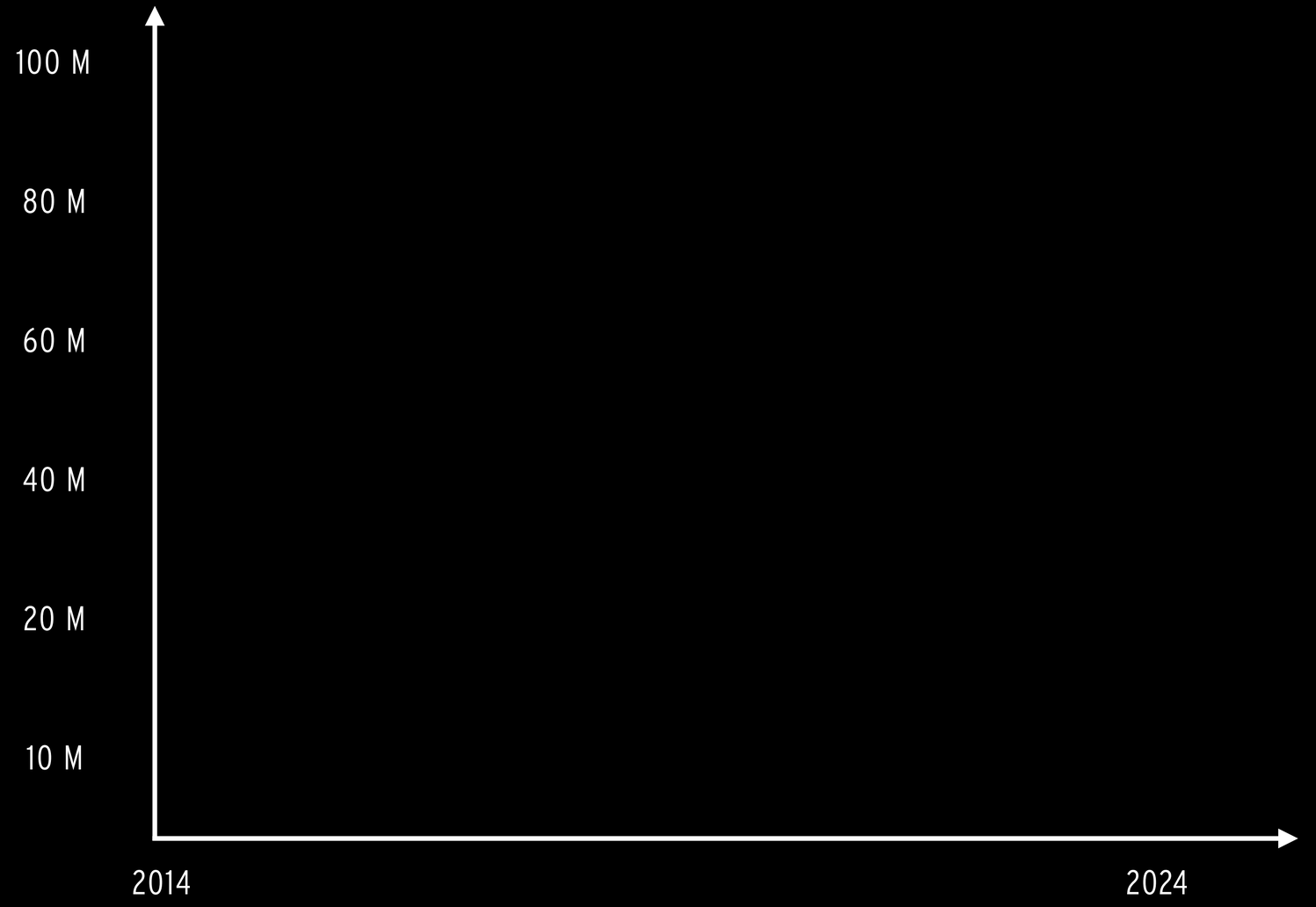
NPM package typosquatting attacks (ongoing)

3CX Desktop App (March 2023)

PyPI package attacks (ongoing)

OPEN SOURCE DEPENDENCIES

How Big is BIG

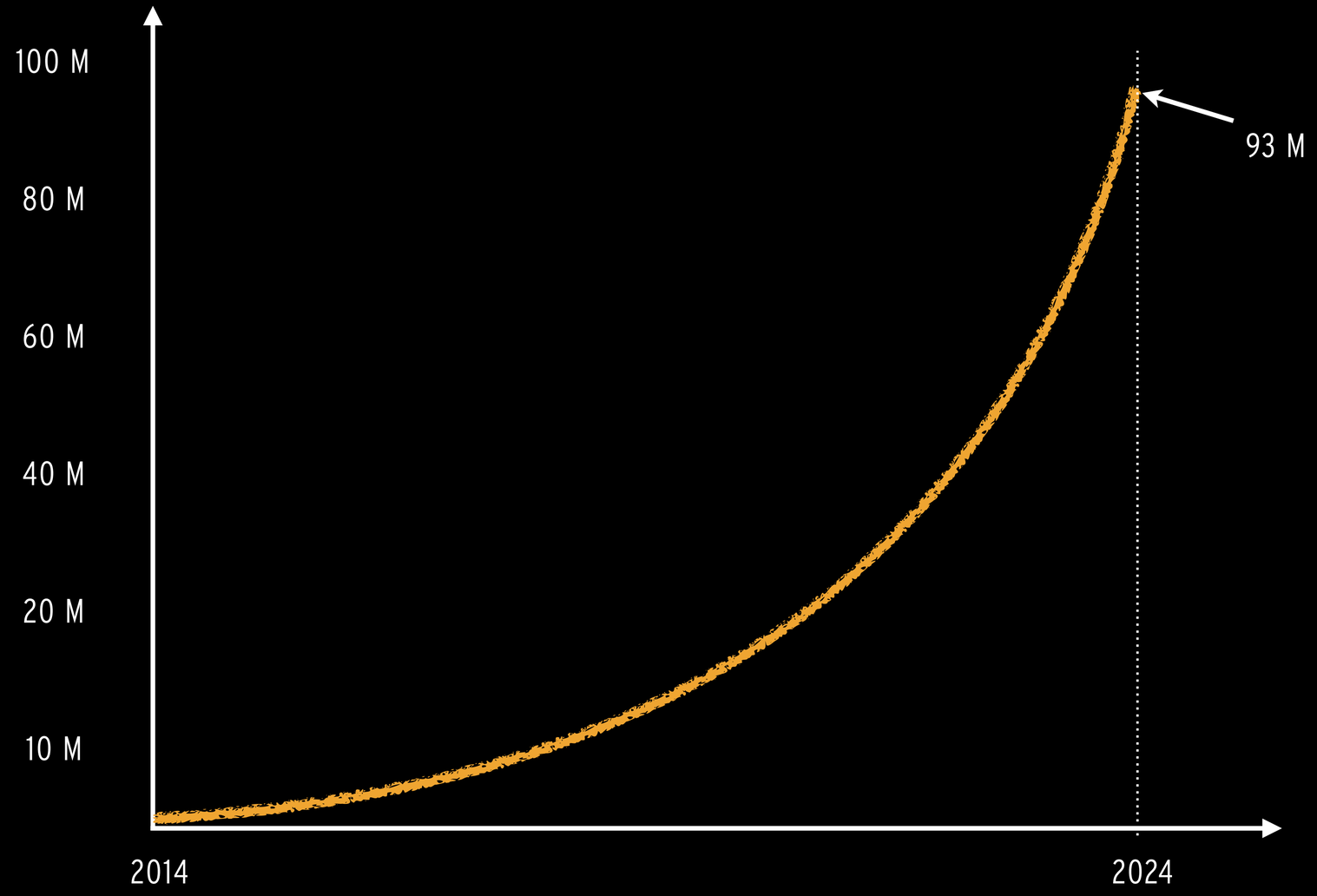


Source: Josh Bressers, CypherCon 7

OPEN SOURCE DEPENDENCIES

How Big is BIG

8.1 million packages, 93 million versioned artifacts



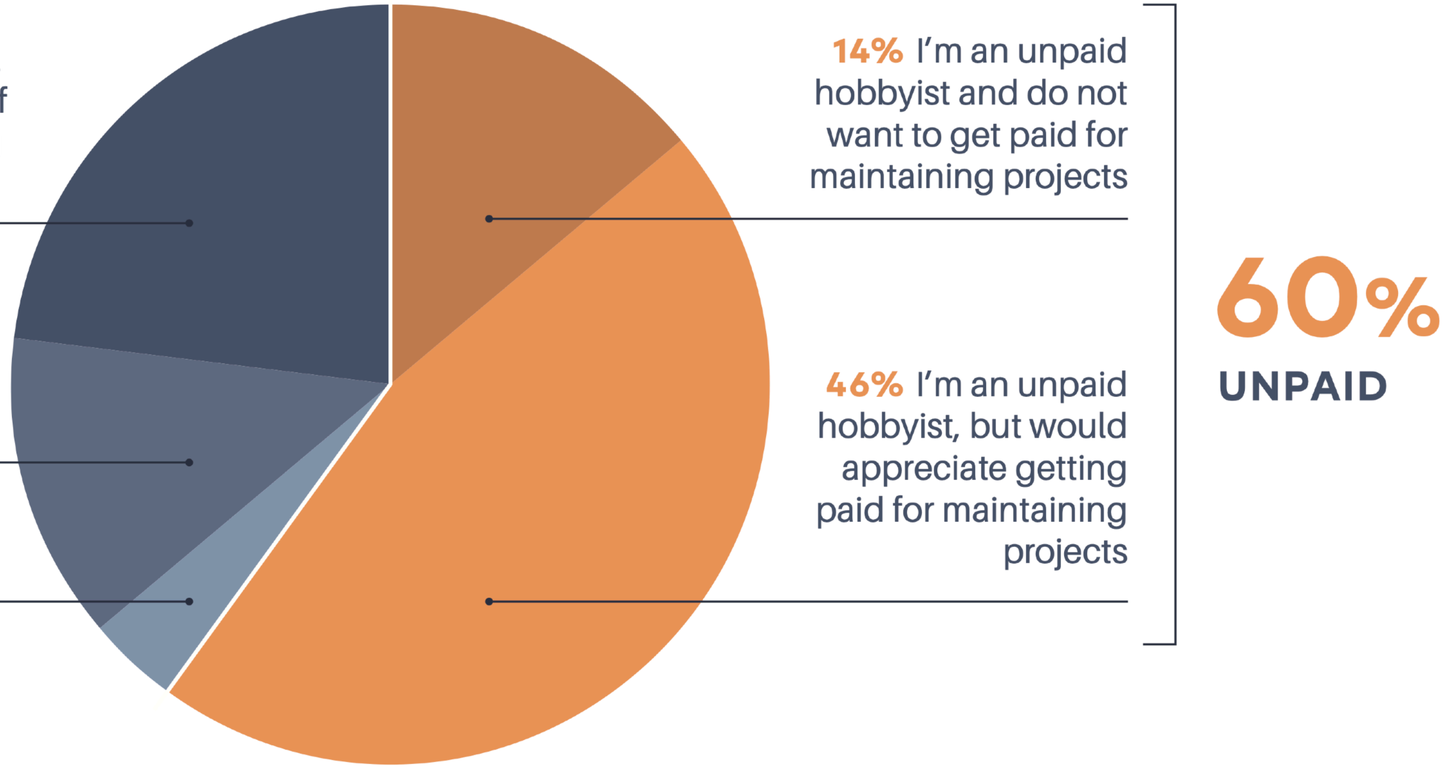
Source: Josh Bressers, CypherCon 7

OPEN SOURCE DEPENDENCIES

An average of 1 maintainer per project (with a few exceptions)

60% of maintainers describe themselves as unpaid hobbyists

Which of the following phrases best describes how you approach your role as an open source maintainer?



n=326

Source: Tidelift, 2023 Open Source Maintainer Report

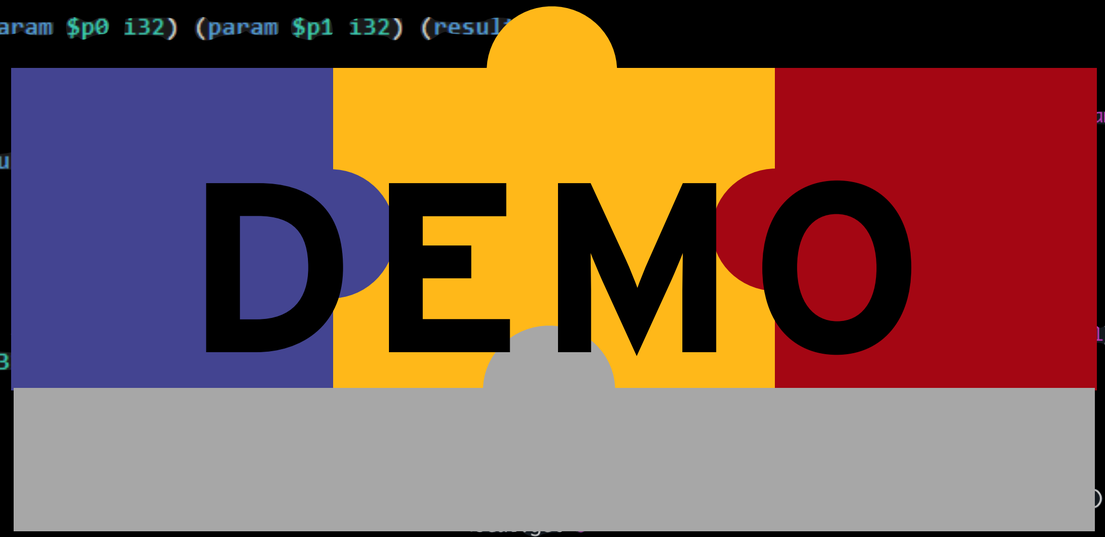
TRUST!

TRUST!

In the code we trust, Really?

Do we have to trust the code?

```
(module
  (type $t0 (func))
  (type $t1 (func (param i32 i32) (result i32)))
  (type $t2 (func (result i32)))
  (func $__wasm_call_ctors (type $t0))
  (func $myAdd (export "myAdd") (type $t1) (param $p0 i32) (param $p1 i32) (resul
    get_local $p1
    get_local $p0
    i32.add)
  (func $main (export "main") (type $t2) (resu
    i32.const 43)
  (table $T0 1 1 anyfunc)
  (memory $memory (export "memory") 2)
  (global $g0 (mut i32) (i32.const 66560))
  (global $__heap_base (export "__heap_base")
  (global $__data_end (export "__data_end") i3
```




```
am.blocking-write-and-flush" (;6;) (type 4) (param i32 i32 i32 i32)

1) (param i32 i32)

) (param i32 i32 i32 i32) (result i32)

    local.get 1
    local.get 2
    local.get 3
    i32.const 8
    call_indirect (type 5)
  )
  (func $adapt-wasi_snapshot_preview1-environ_get (;9;) (type 6) (param i32 i32) (result i32)
    local.get 0
    local.get 1
    i32.const 9
    call_indirect (type 6)
  )
  (func $adapt-wasi_snapshot_preview1-environ_sizes_get (;10;) (type 6) (param i32 i32) (result i32)
    local.get 0
    local.get 1
    i32.const 10
    call_indirect (type 6)
  )
```

```
(module
  (type $t0 (func))
  (type $t1 (func (param i32 i32) (result i32)))
  (type $t2 (func (result i32)))
  (func $__wasm_call_ctors (type $t0))
  (func $myAdd (export "myAdd") (type $t1) (param $p0 i32) (param $p1 i32) (result i32)
    get_local $p1
    get_local $p0
    i32.add)
  (func $main (export "main") (type $t2) (result i32)
    i32.const 43)
  (table $T0 1 1 anyfunc)
  (memory $memory (export "memory") 2)
  (global $g0 (mut i32) (i32.const 66560))
  (global $__heap_base (export "__heap_base") i32 (i32.const 66560))
  (global $__data_end (export "__data_end") i32 (i32.const 1024))
)
```

```
(func $"indirect-wasi:io/streams@0.2.2-[method]output-stream.blocking-write-and-flush" (;6;) (type 4) (param i32 i32 i32 i32)
  local.get 0
  local.get 1
  local.get 2
  local.get 3
  i32.const 6
  call_indirect (type 4)
)
(func $indirect-product:service/api@0.1.0-get (;7;) (type 1) (param i32 i32)
  local.get 0
  local.get 1
  i32.const 7
  call_indirect (type 1)
)
(func $adapt-wasi_snapshot_preview1-fd_write (;8;) (type 5) (param i32 i32 i32 i32) (result i32)
  local.get 0
  local.get 1
  local.get 2
  local.get 3
  i32.const 8
  call_indirect (type 5)
)
(func $adapt-wasi_snapshot_preview1-environ_get (;9;) (type 6) (param i32 i32) (result i32)
  local.get 0
  local.get 1
  i32.const 9
  call_indirect (type 6)
)
(func $adapt-wasi_snapshot_preview1-environ_sizes_get (;10;) (type 6) (param i32 i32) (result i32)
  local.get 0
  local.get 1
  i32.const 10
  call_indirect (type 6)
)
```

DEMO – TOOLS

cargo	Rust package manager
wit-bindgen	WIT Language binding tool (code generator)
wasm-tools	A collection of tools (sub-commands) for working with wasm modules and components
wac	Web Assembly Component Composition tool
wkg	Package tool for fetching and publishing Wasm Components to OCI or Warg registries
wasmtime, spin	Standalone WASM Runtimes (VM)
docker	Wasm containerd runtimes (shim)
teavm, maven	AOT Java Bytecode to JS & WASM Compiler (Fermyon fork with WASI support)

... and good old make to run them all

WASI/WASM BACKEND RUNTIMES

Standalone runtimes	wasmtime, wasmedge, wasmer, spin, node, …
Docker/containerd	Runtime shims from Bytecode Alliance, WasmEdge, Spin, Slight, Wasm Workers Server, Lunatic, Wasmer
OpenShift/K8s/CRI-O	crun-wasm enabled worker nodes (MachineConfig, RuntimeClass)
NGINX Unit	WASI-HTTP modules
Cloud Native Apps	Fermyon Spin, Wasm Workers Server, wasmCloud, wasmer.io, fastly, MoonBit, …

WASI/WASM ROADMAP – INTERESTING FEATURES

	Now	Next	Later (? years)
Core WASM (WG)	Release 2.0	Release 3.0 Core Threads Exception handling Memory64	~ 40 proposals in the pipeline memory model, security etc
Component Model & WIT	Preview 2	Preview 3 GC in Components Native Async	Component Model 1.0
WASI	Preview 2	Preview 3 Native Async Futures and Streams	WASI 1.0

SUMMARY

WASM, Component Model & WASI

is definitely an interesting and promising backend technology

targets real issues dealing with unsafe code

backend apps requires Component Model and WIT

in production use and ready for certain use-cases and platforms

SUMMARY

But...

the ecosystem might be too complex for developers to embrace

slowly paving its way through the standardization process

needs improved language and tooling support to become universal

not yet ready for JVM based languages

SUMMARY

LINKS

- https://www.w3.org/TR/wasm-core-2/
- https://www.javaadvent.com/2024/12/wasm-4-the-java-geek-3-electric-boogaloo.html
- https://bytecodealliance.org/articles/webassembly-the-updated-roadmap-for-developers
- https://component-model.bytecodealliance.org/tutorial.html
- https://github.com/appcypher/awesome-wasm-langs
- https://github.com/mbasso/awesome-wasm
- https://github.com/mcuking/Awesome-WebAssembly-Applications
- https://www.redhat.com/en/blog/webassembly-wasm-and-openshift-a-powerful-duo-for-modern-applications
- https://www.docker.com/blog/wasm-vs-docker/