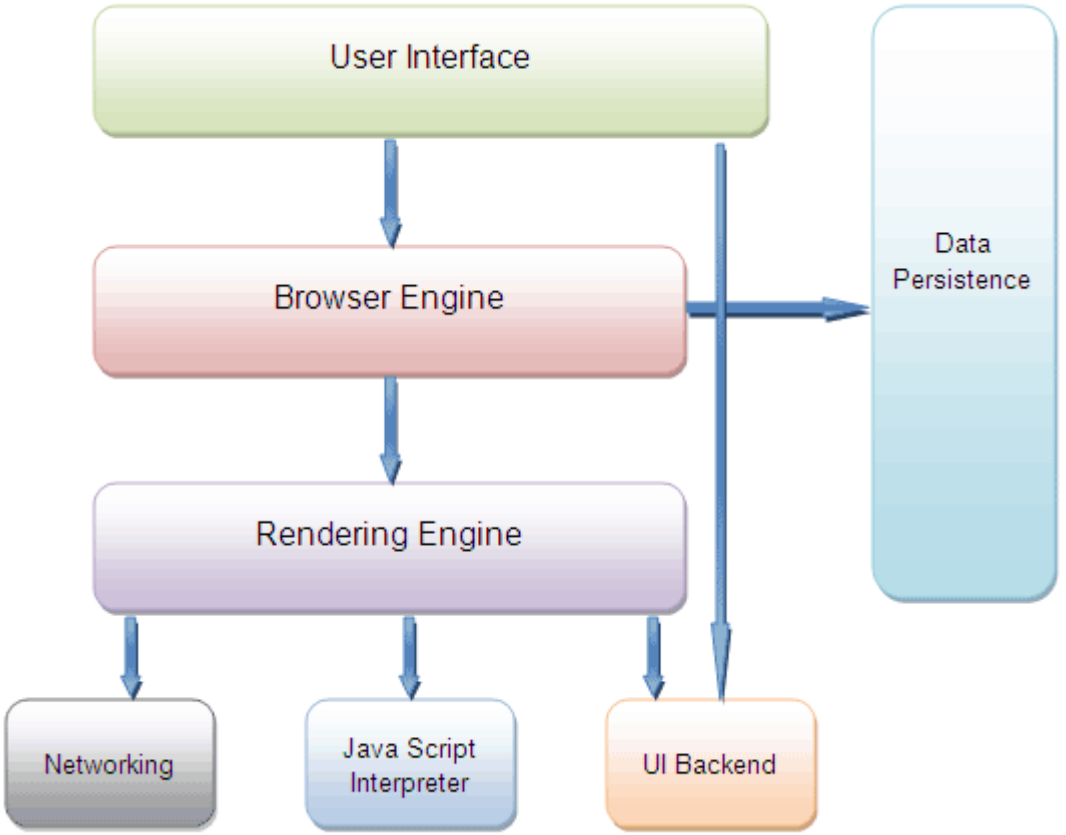
WASM
(WebAssembly)

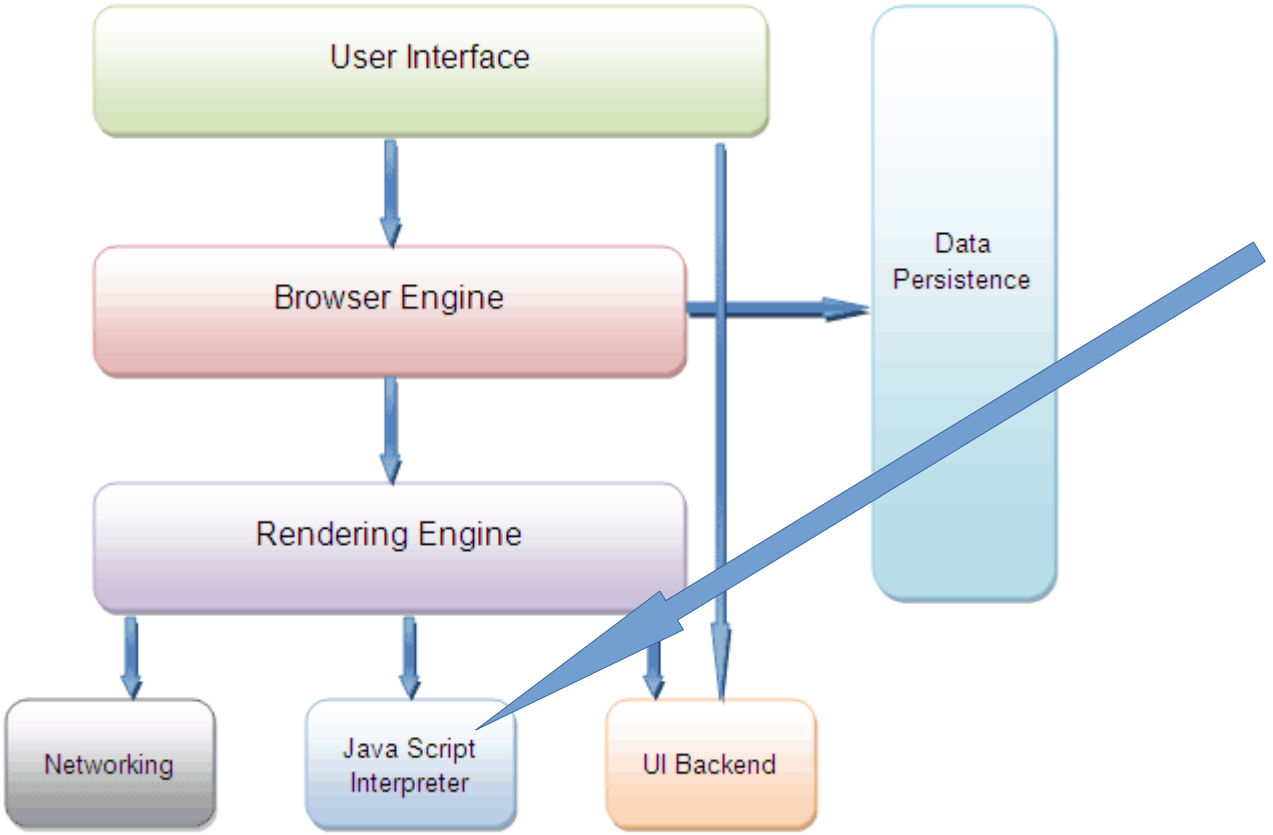
KWLUG March 2022

(another talk with many logos)

Origin story: Browser



Origin story: Browser

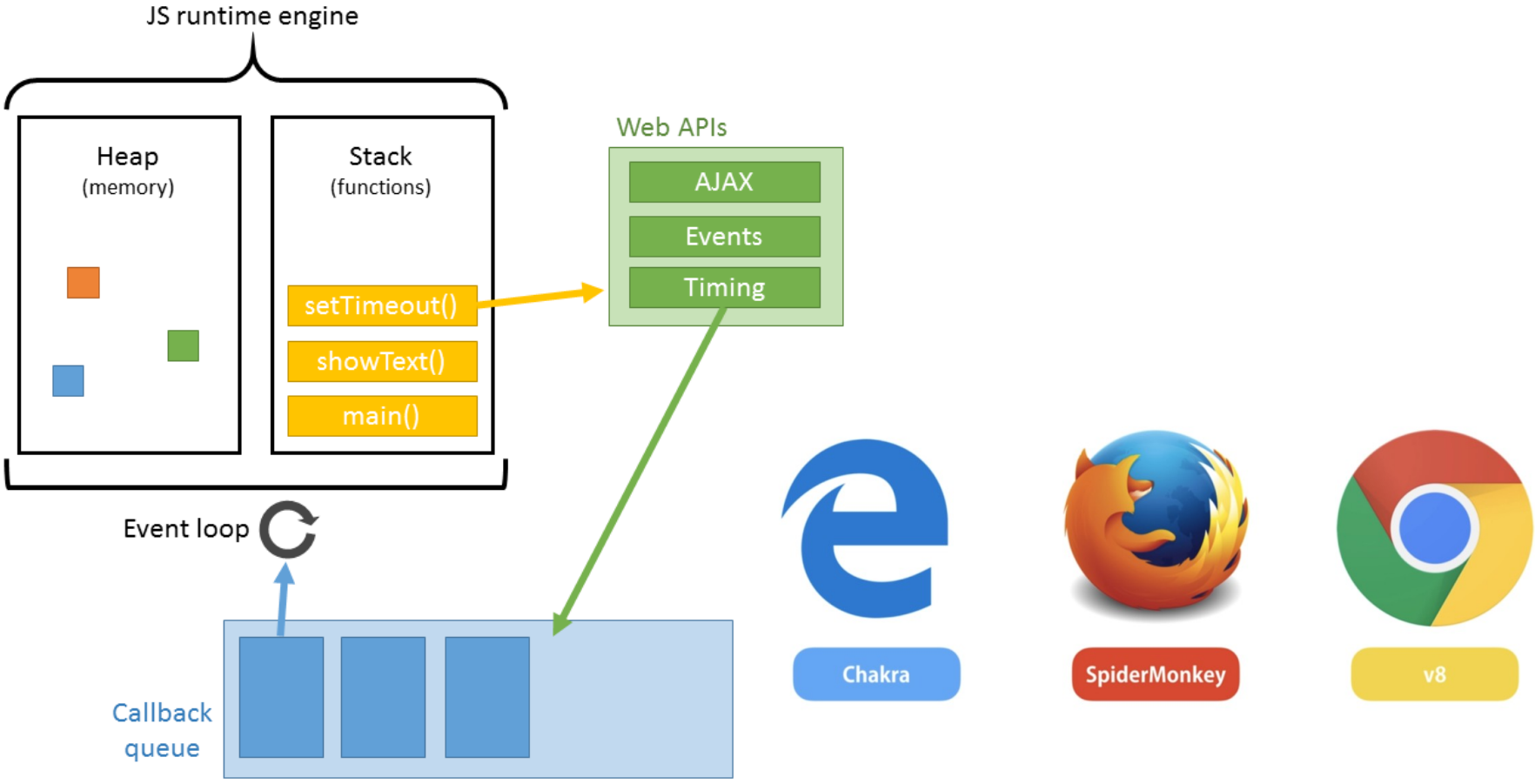


JS

Place of JVM,

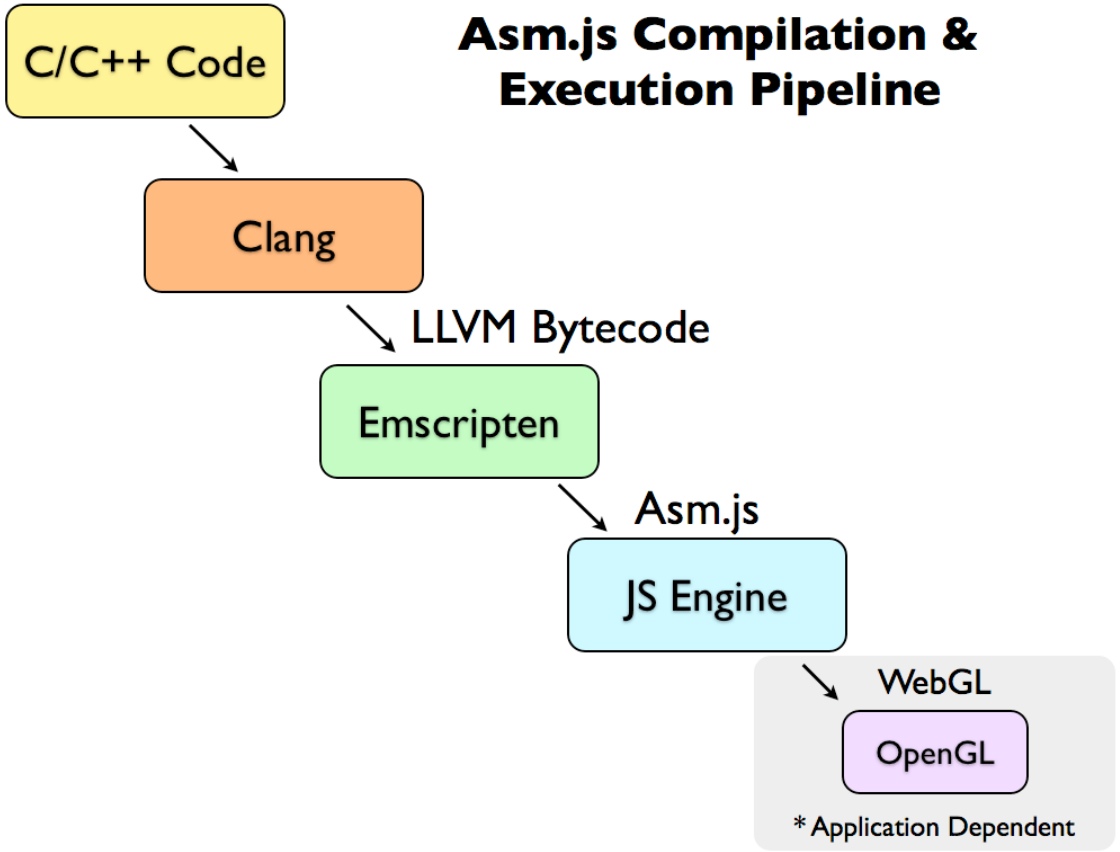
Process Virtual Machine
vs
System Virtual Machine

Origin story: JavaScript runtime/vm



Origin story: asm.js as a compilation target

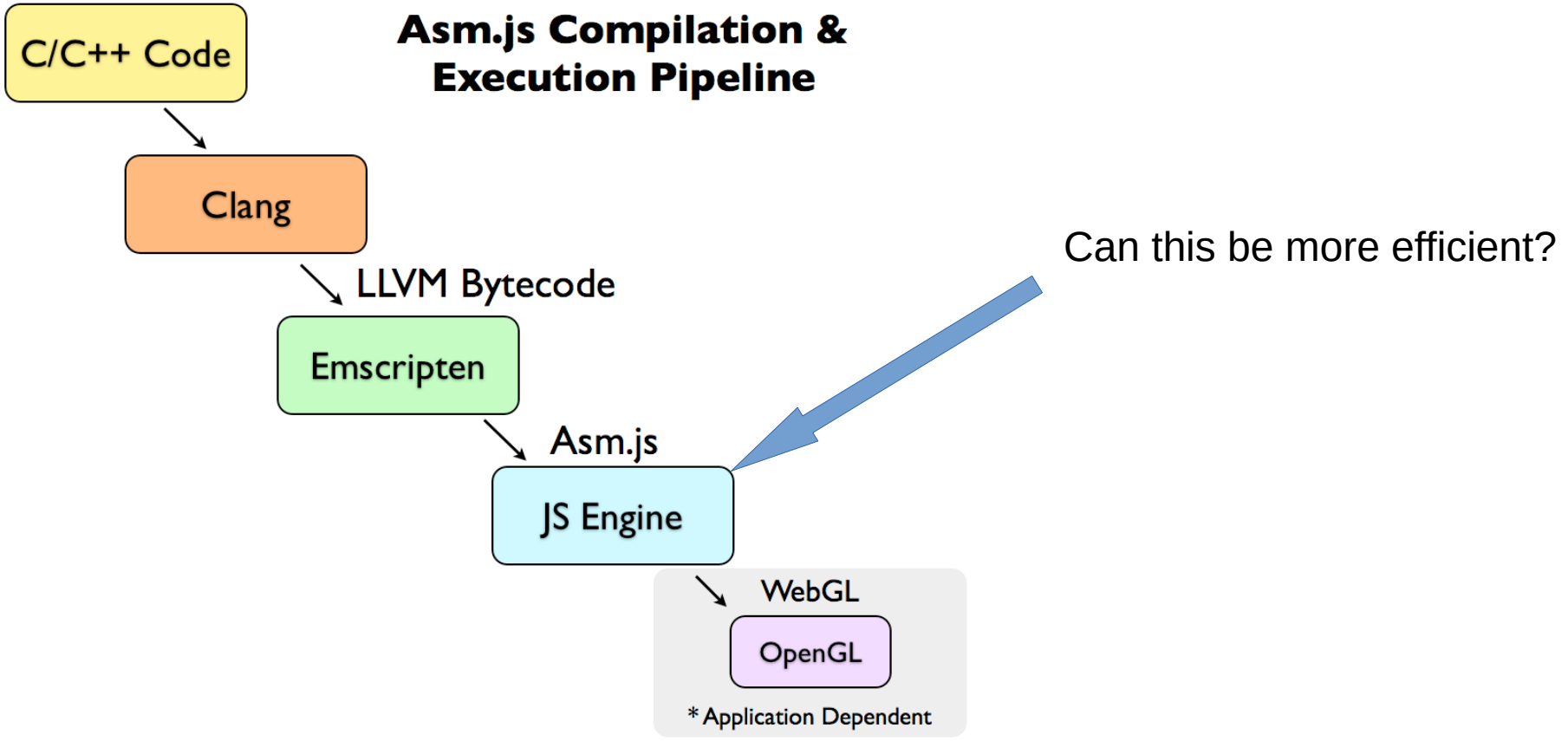
(efficiency of simplified input in performance tuned vm)



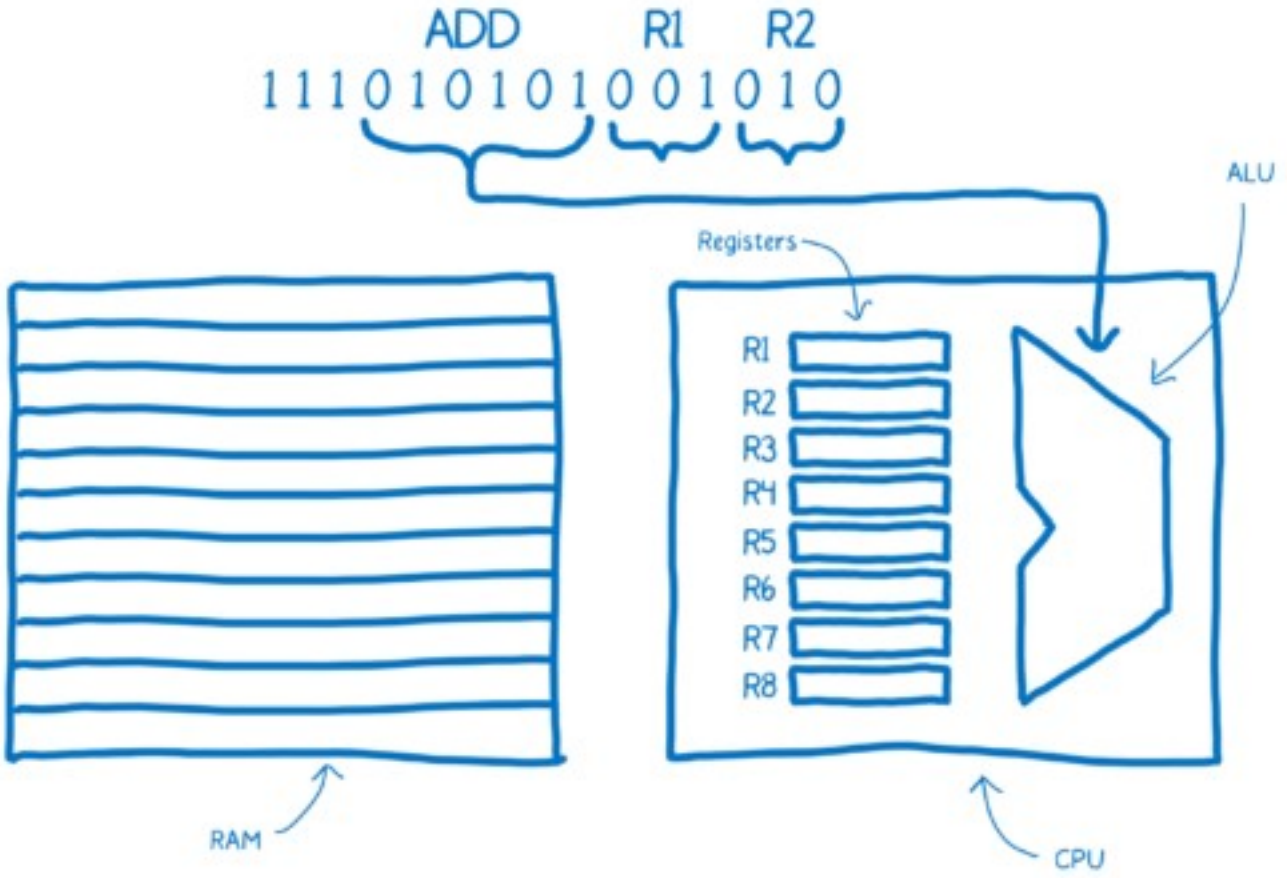

```
int f(int i) {
  return i + 1;
}
```

```
function f(i) {
  i = i|0;
  return (i + 1)|0;
}
```

Origin story: from asm.js to WebAssembly



WebAssembly stack machine



https://hacks.mozilla.org/2017/02/a-crash-course-in-assembly/

WebAssembly is a virtual instruction set architecture (virtual ISA)

```
(module
 (type $i32_i32_⇒_i32 (func (param i32 i32) (result i32)))
 (memory $0 0)
 (export "calculate" (func $module/calculate))
 (export "memory" (memory $0))
 (func $module/calculate (param $0 i32) (param $1 i32) (result i32)
  local.get $0
  local.get $1
  i32.mul
  i32.const 5
  i32.add
 )
)
```


```
export function calculate(a: i32, b: i32): i32 {
  let c = a * b;
  c += 5;
  return c;
}
```

https://www.assemblyscript.org/

Embedding in JavaScript runtime

▾ Object reference

- `WebAssembly`
- `WebAssembly.Module`
- `WebAssembly.Global`
- `WebAssembly.Instance`
- `WebAssembly.Memory`
- `WebAssembly.Table`
- `WebAssembly.CompileError`
- `WebAssembly.LinkError`
- `WebAssembly.RuntimeError`

https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts

Demo in browser, on
https://www.assemblyscript.org/

WASM compile target in Rust, wasm-pack and wasm-bindgen

wasm-pack: install following https://rustwasm.github.io/

wasm-bindgen code docs: https://rustwasm.github.io/wasm-bindgen/

Message passing to and from WASM

Code used in 3NWeb client:

Rust side of binding (message passing):
https://github.com/3nsoft/wasm-message-passing-3nweb/blob/main/src/wasm_mp1.rs

Rust code in WASM:
https://github.com/3nsoft/core-3nweb-client-lib/blob/master/wasm-parts/cryptor/src/lib.rs

Embedding node.js side of binding (message passing):
https://github.com/3nsoft/core-3nweb-client-lib/blob/master/ts-code/lib-client/cryptor/wasm-mp1-modules.ts

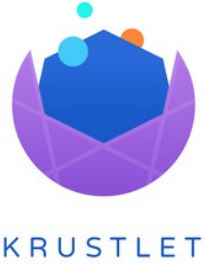
Usage by embedding node.js side:
https://github.com/3nsoft/core-3nweb-client-lib/blob/master/ts-code/lib-client/cryptor/worker-wasm.ts

WASI: WebAssembly system interface

WASI: https://wasi.dev/

wasmtime:
https://docs.wasmtime.dev/

krustlet – running Kubernetes WASM workloads:
https://docs.krustlet.dev/howto/wasm/



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!

> **Lin Clark** @linclark · Mar 27, 2019
> WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...
>
> 📢 Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)
>
> hacks.mozilla.org/2019/03/standa...
> Show this thread

4:39 PM · Mar 27, 2019 · Twitter Web Client