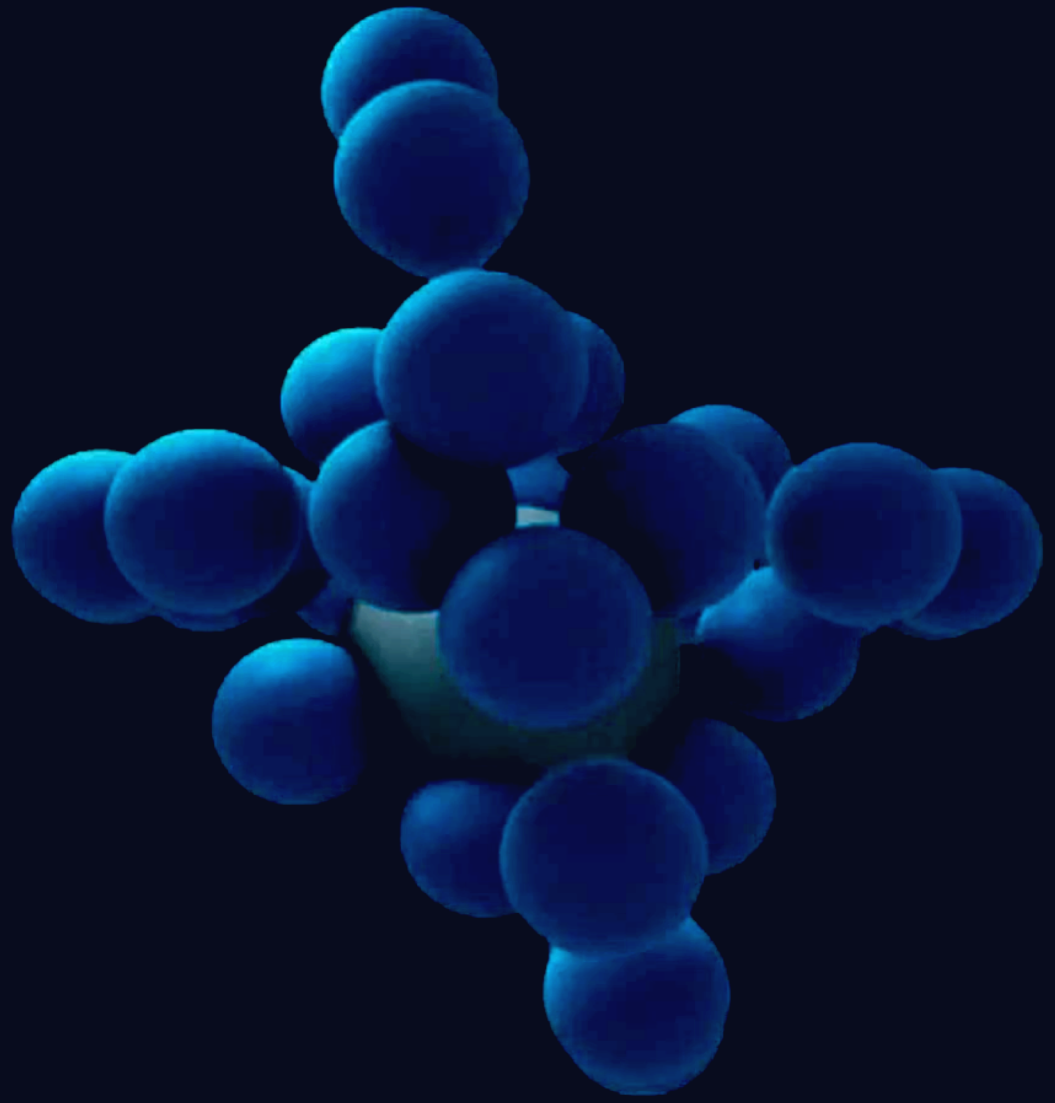
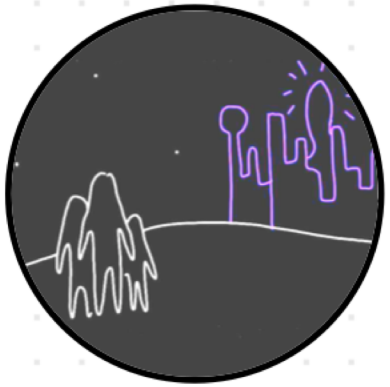
JavaScript and WebAssembly: story of a symbiosis



> you

Developers (web)

> me

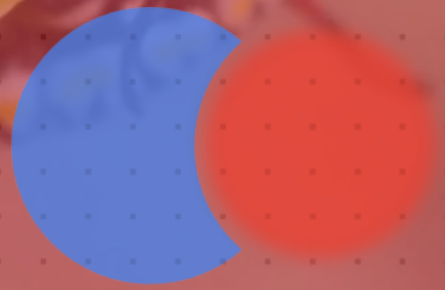


Wasm Nano Handbook + AS

@maudnals

Berlin Wasm Meetup

> symbiosis



Parasitism

Mutualism

> agenda

*Wasm = WebAssembly

JS, Wasm* and the browser

Symbiosis: How JS and Wasm benefit from each other

MICRO/MACRO

Symbiosis: Writing Wasm in TypeScript

META

Demo

Go

QnA

JS, Wasm and the browser

> JS

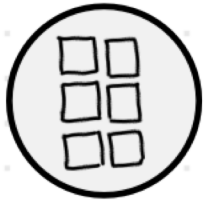
Web and more

High-level

Dynamically typed

> Wasm

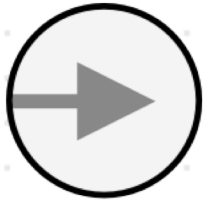
New language for the web



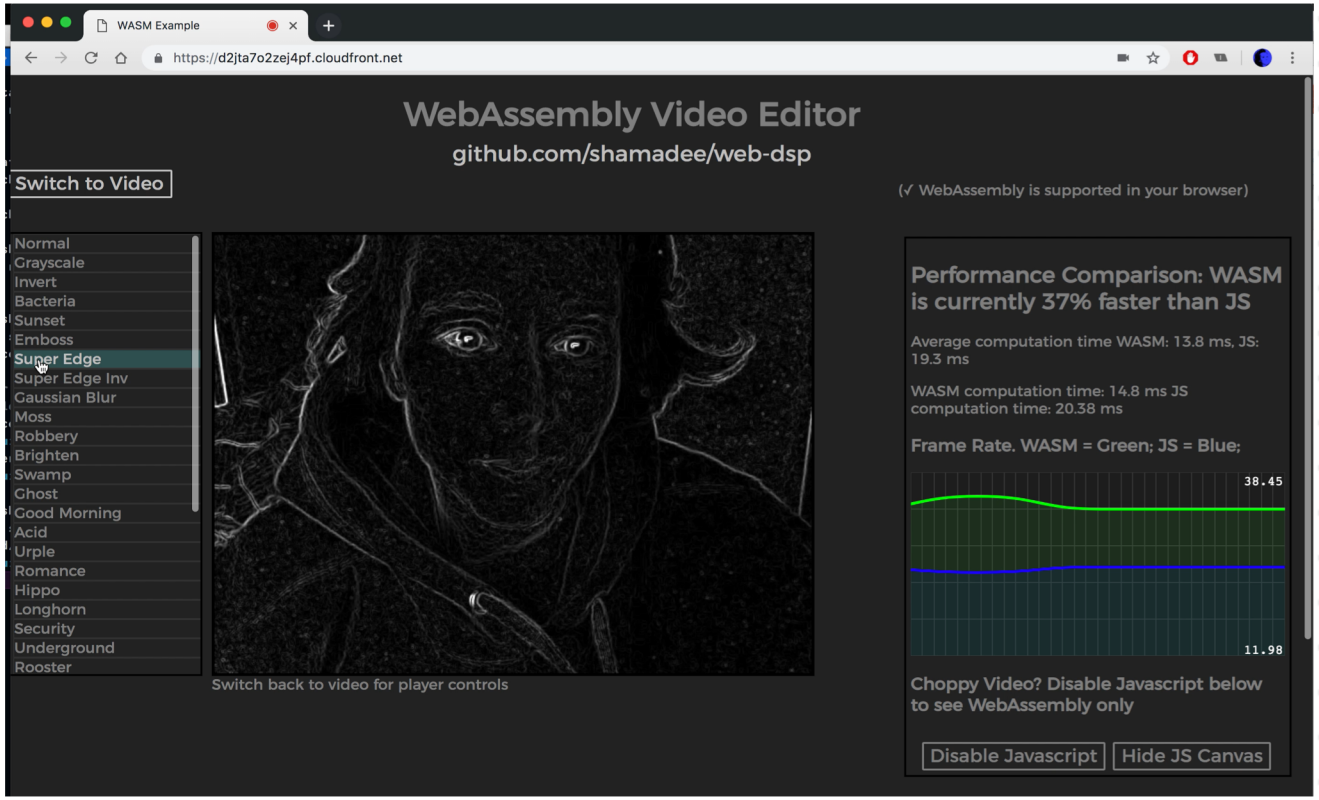
Low-level bytecode

Typed

Near-native speed, compact binary format



Compilation target

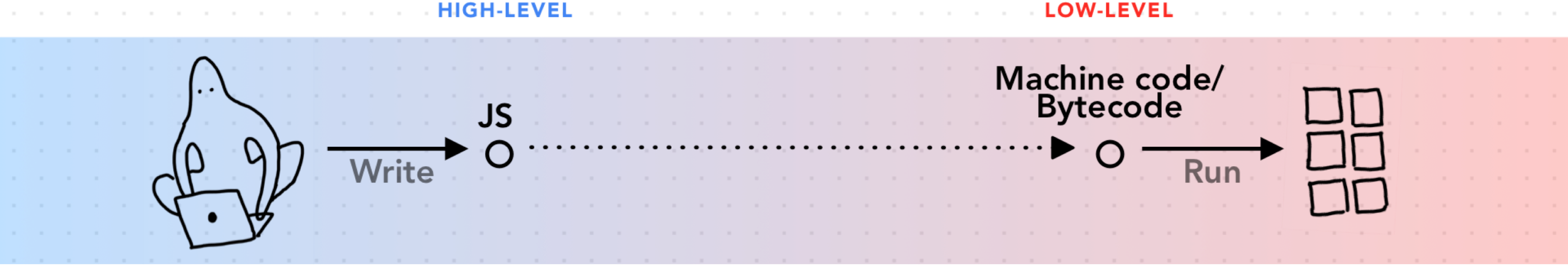


@maudnals
04.19

> Wasm

```
(module
  (func $add (param $lhs i32)
  (param $rhs i32) (result i32)
    get_local $lhs
    get_local $rhs
    i32.add)
  (export "add" (func $add))
)
```

> human
to metal



> human
to metal

HIGH-LEVEL

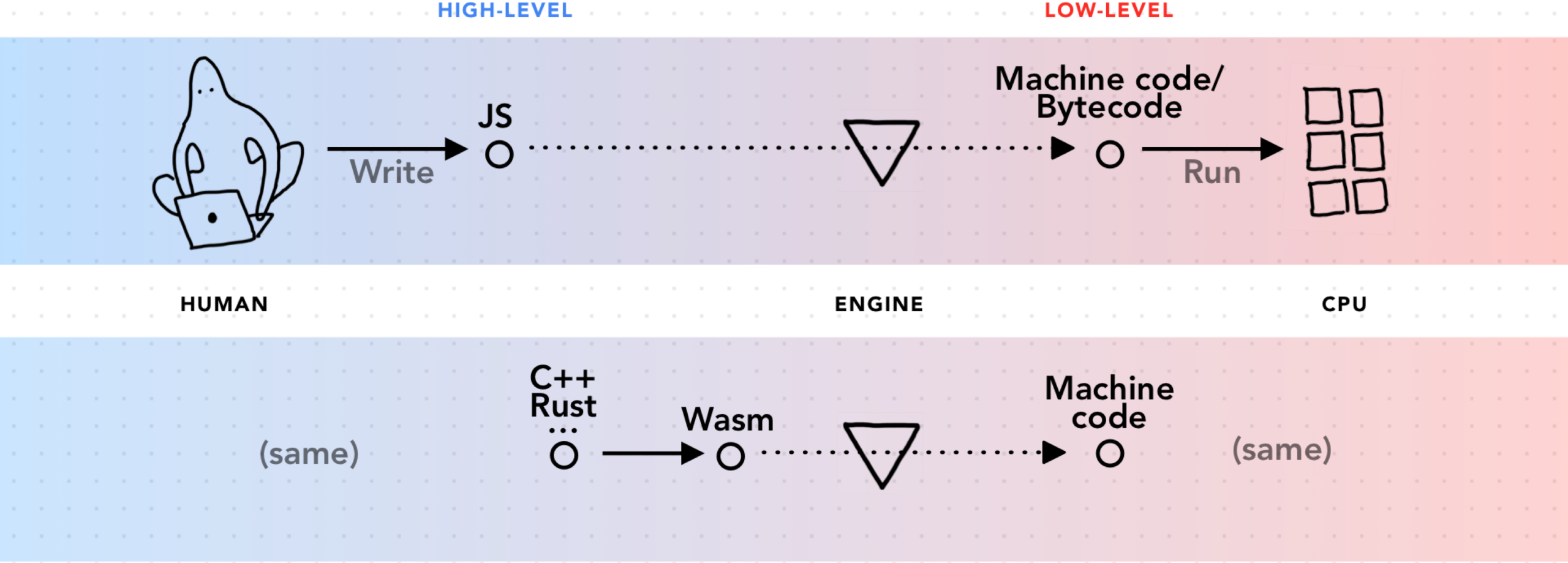
LOW-LEVEL

HUMAN

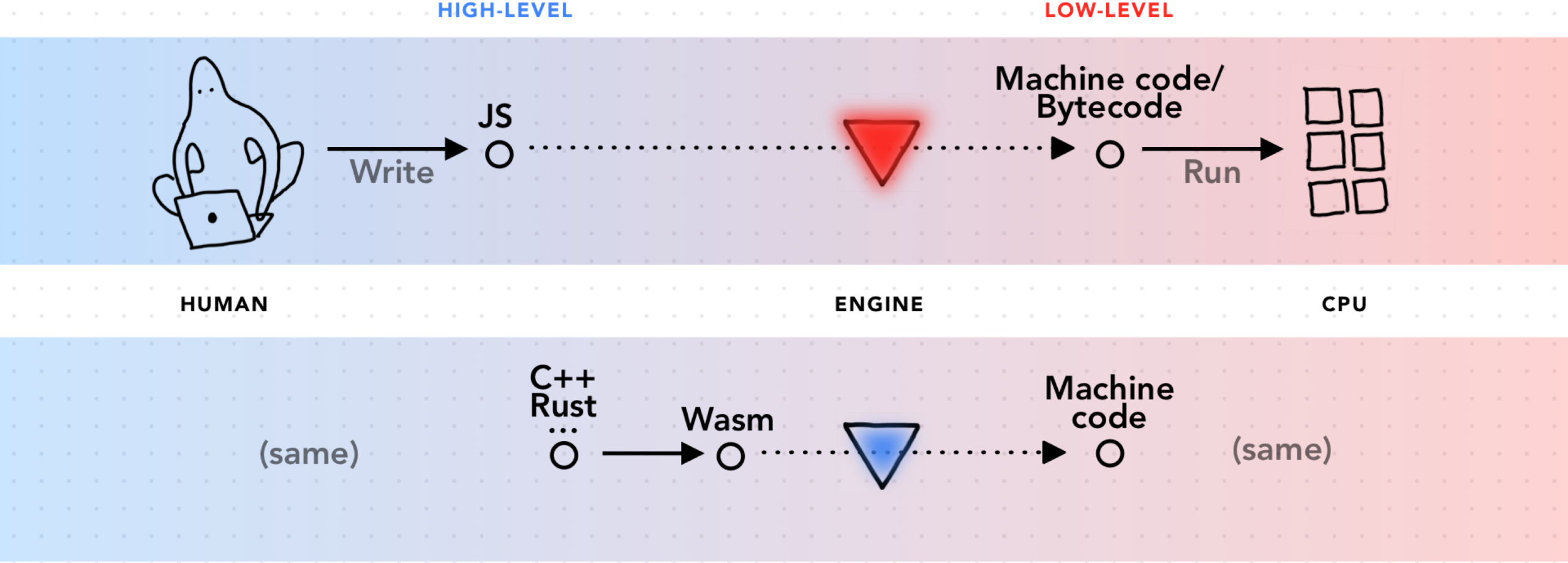
ENGINE

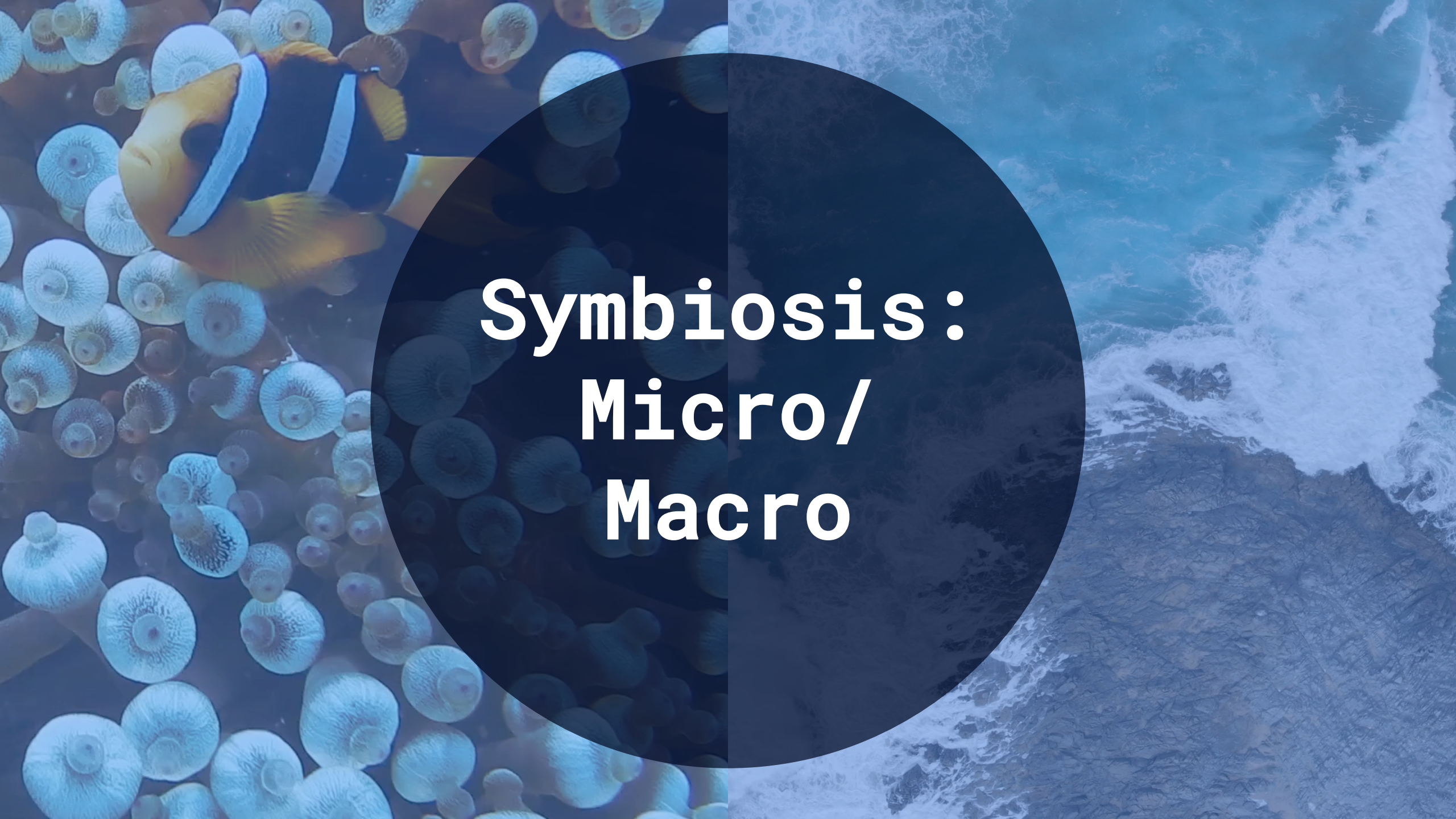
CPU

> human
to metal



> human
to metal





Symbiosis: Micro/ Macro

> interop

JS index.js

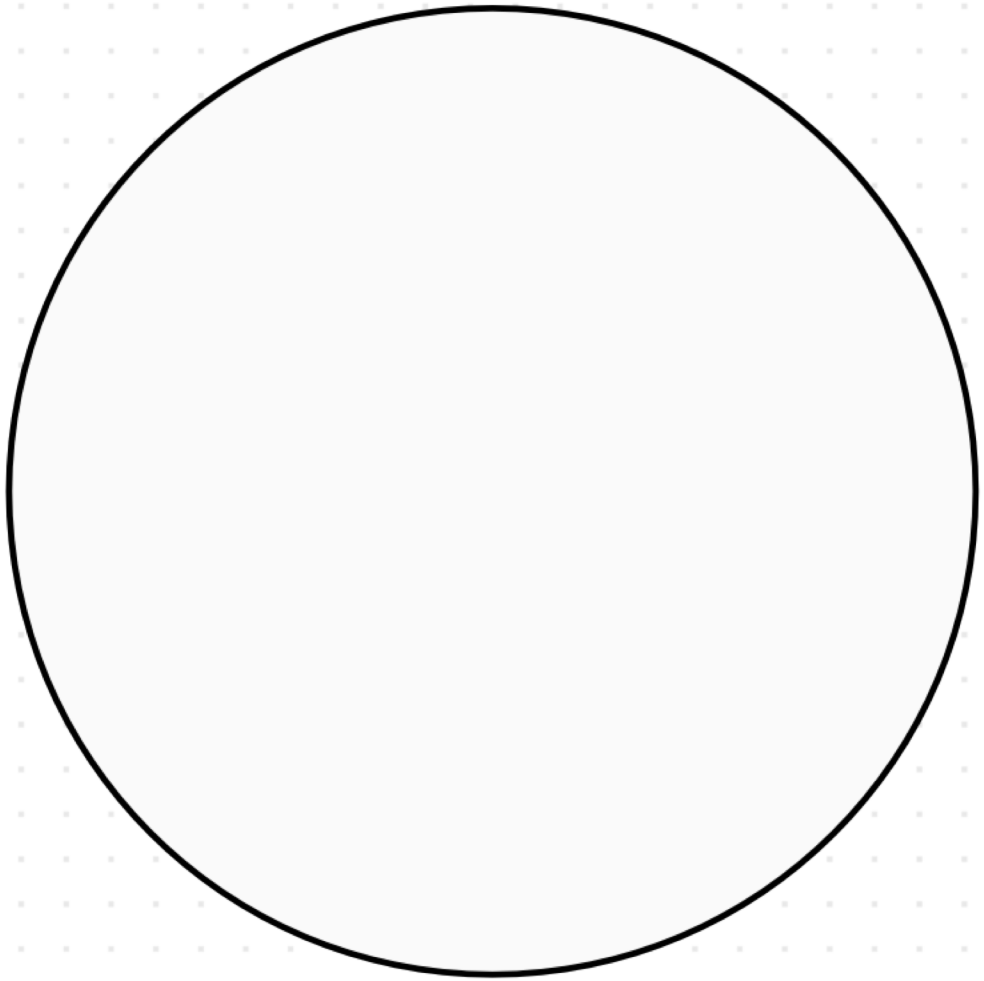
```
const importObject = {
  console: {
    log: function(arg) {
      console.log(arg);
    }
  }
};

WebAssembly.instantiateStreaming(
  fetch('x.wasm'), importObject)
  .then(obj => {
    obj.instance.exports.log(); // 42
  });
```

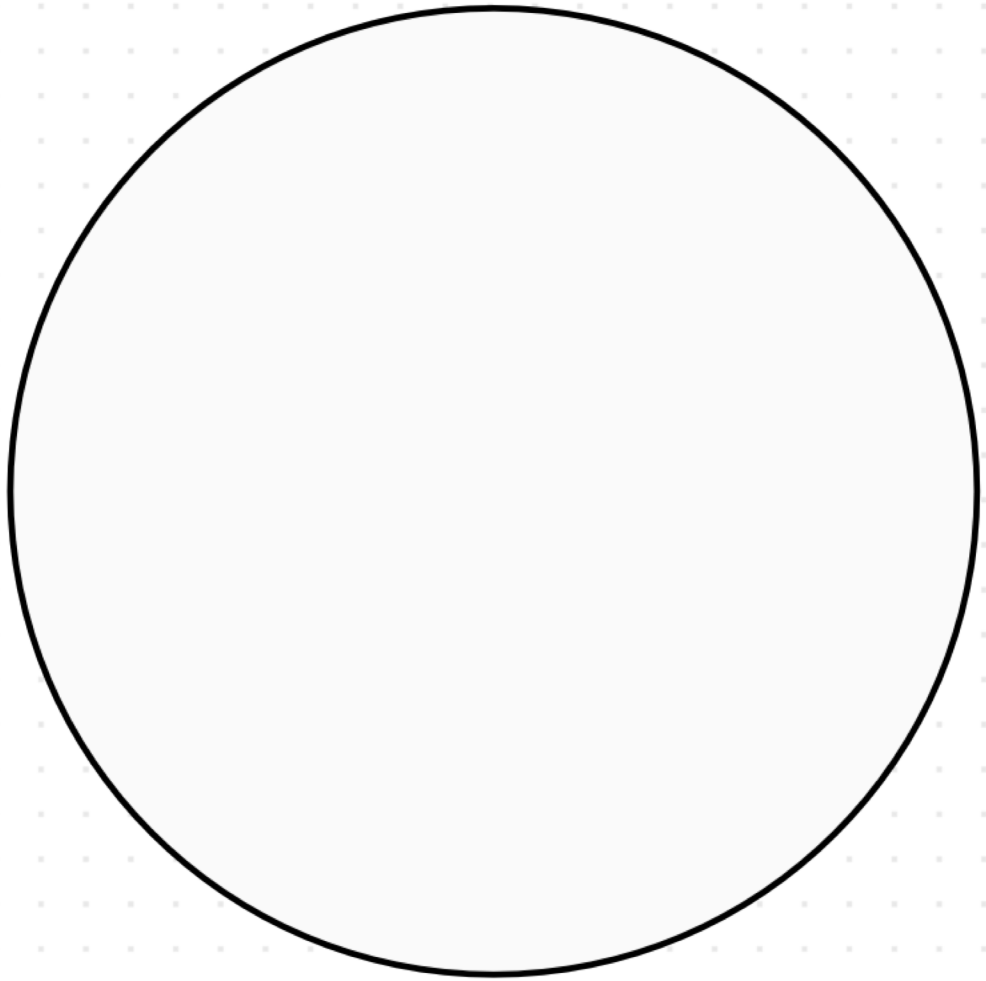
WA x.wasm (wat)

```
(module
  (import "console" "log" (func $log
  (param i32)))
  (func (export "log")
    i32.const 42
    call $log))
```

> benefits:
micro

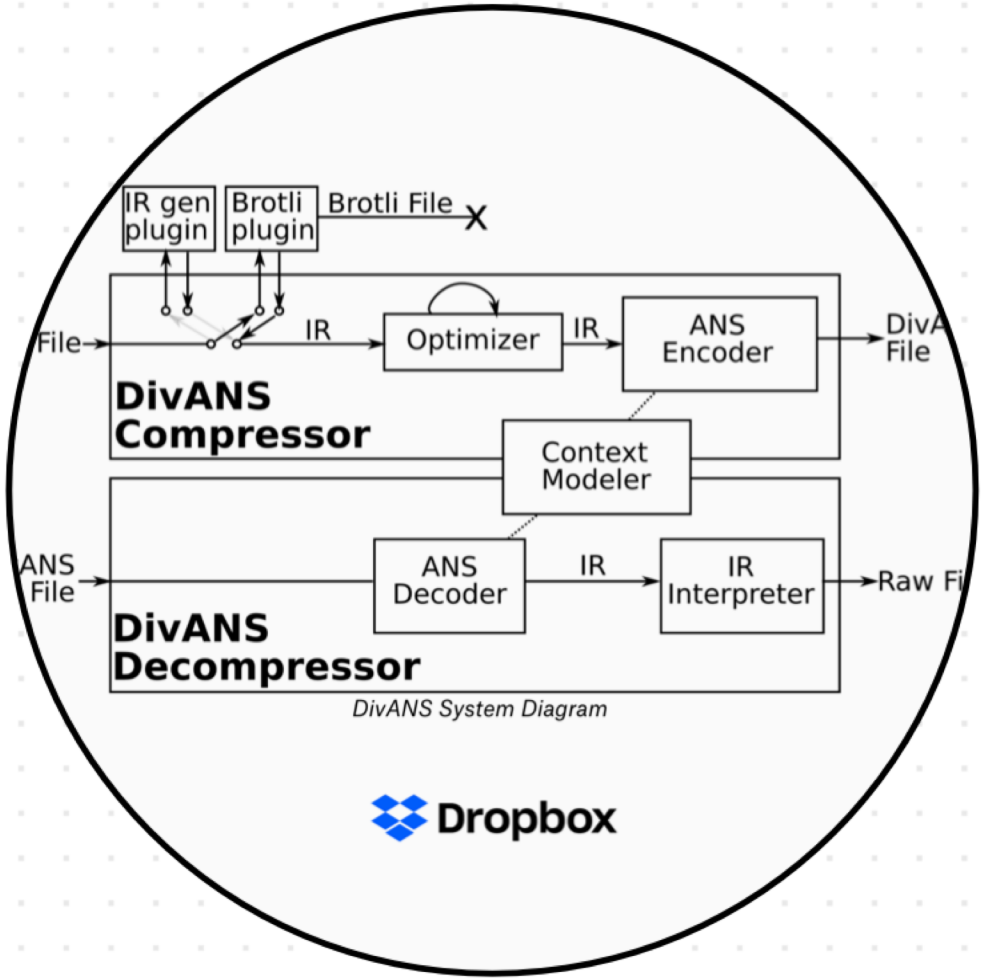


Wasm for JS



JS for Wasm

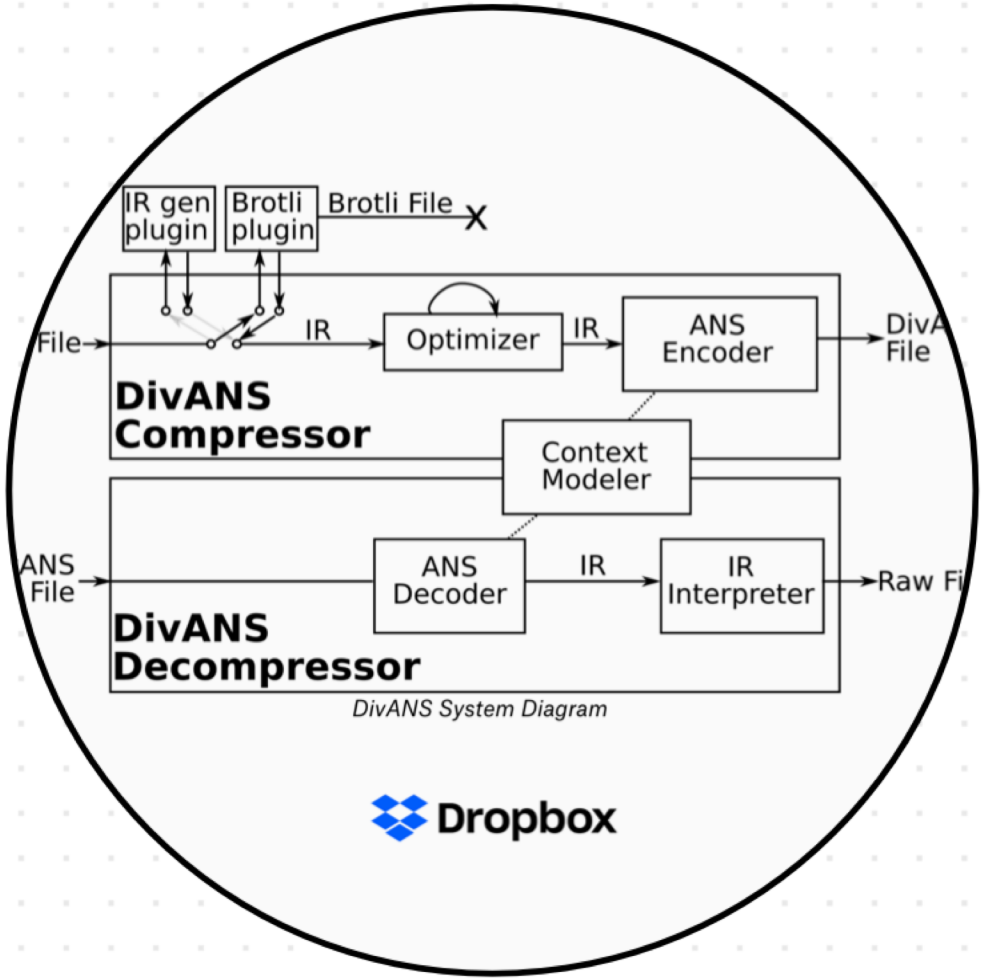
> benefits:
micro



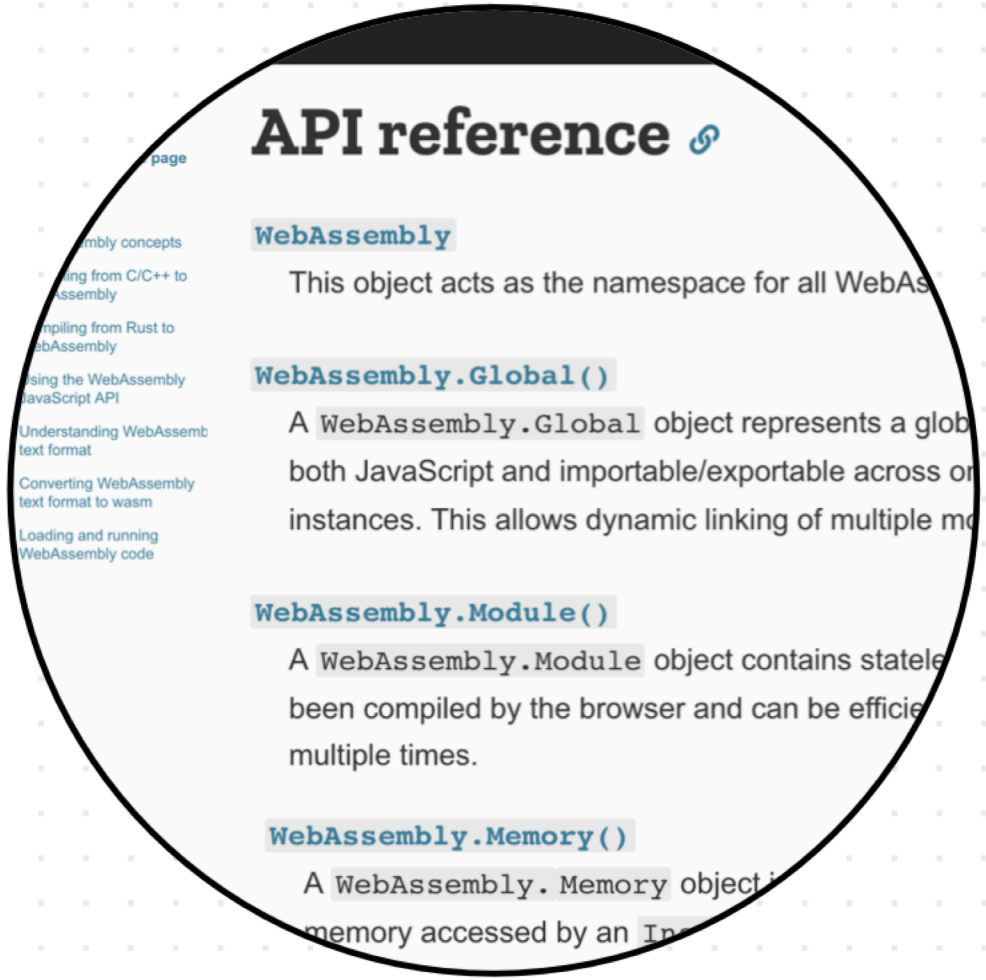
Wasm for JS

JS for Wasm

> benefits:
micro

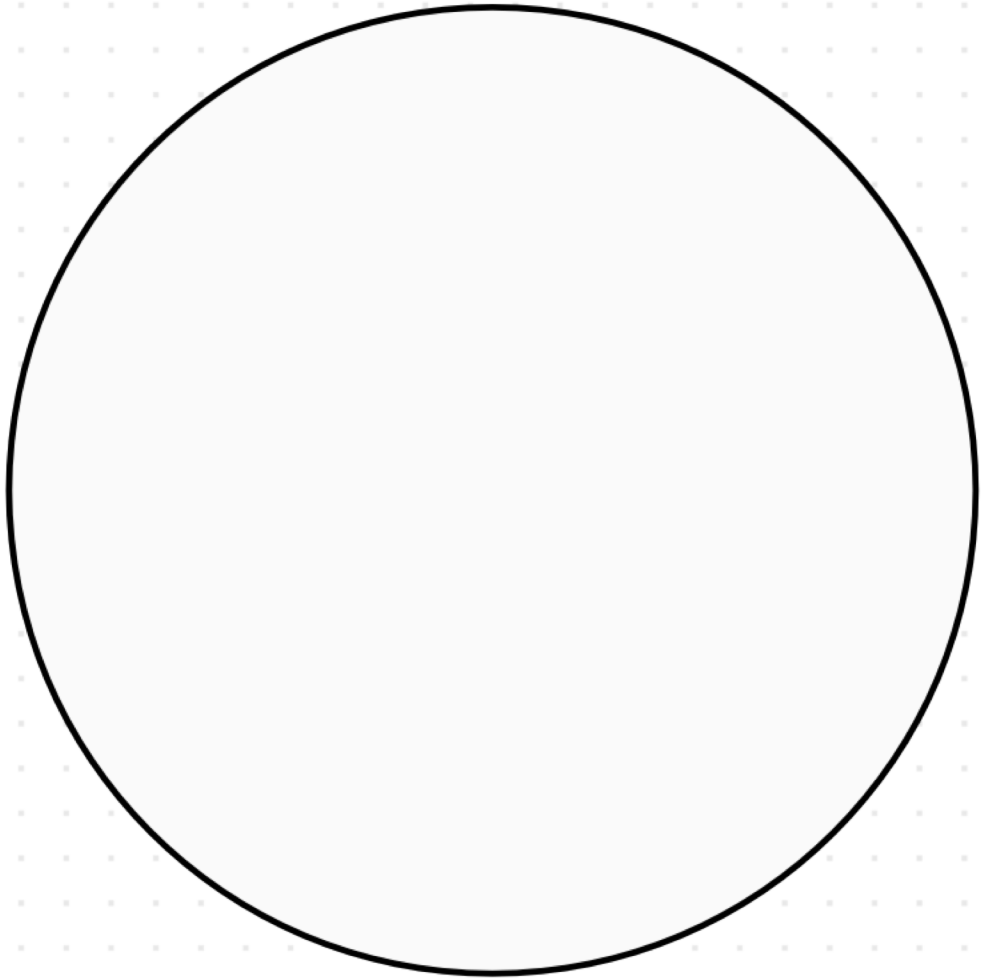


Wasm for JS

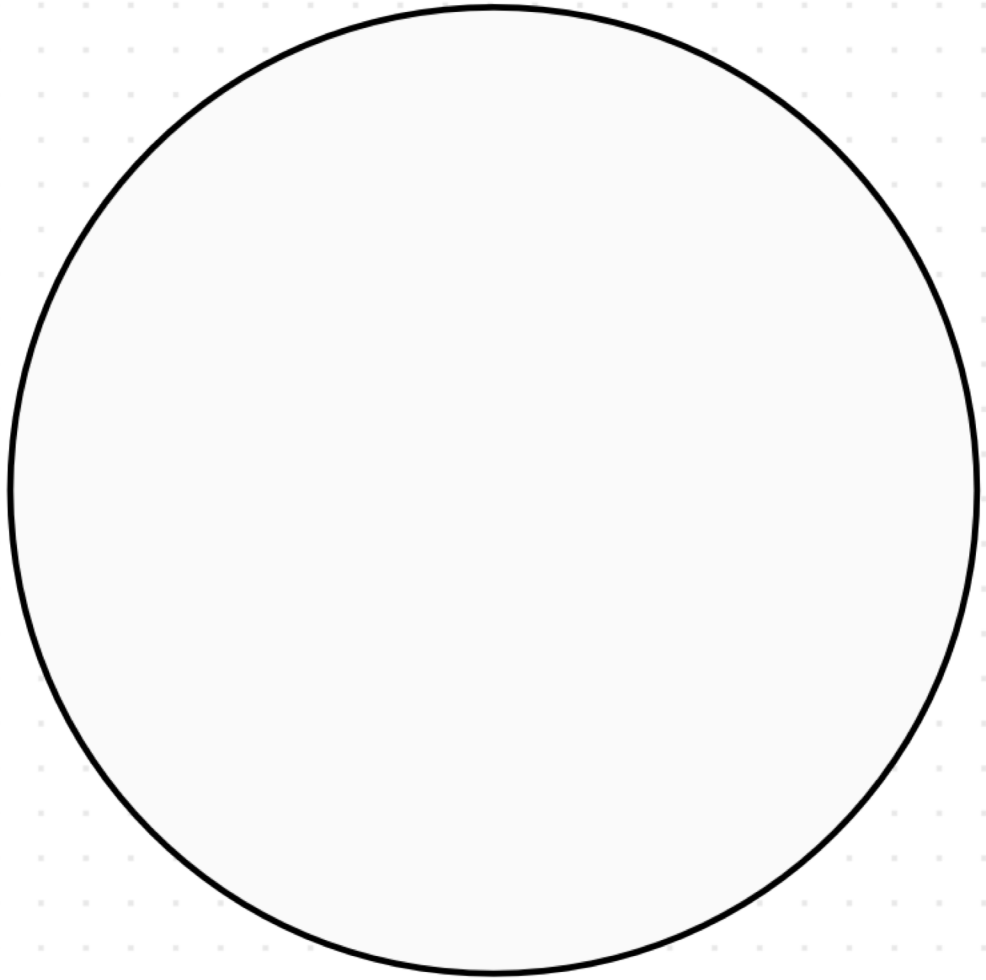


JS for Wasm

> benefits:
macro



Wasm for JS



JS for Wasm

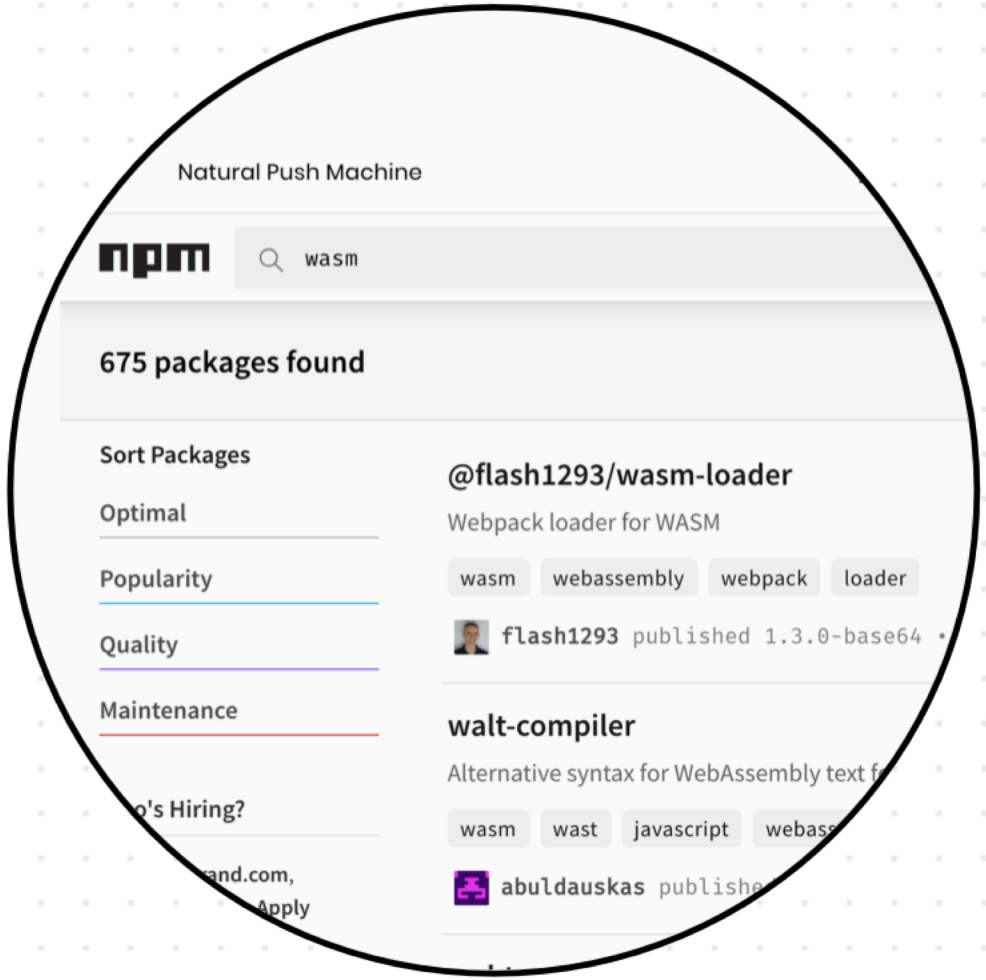
> benefits:
macro

Wasm for JS

JS for Wasm

> benefits:
macro

Wasm for JS



JS for Wasm

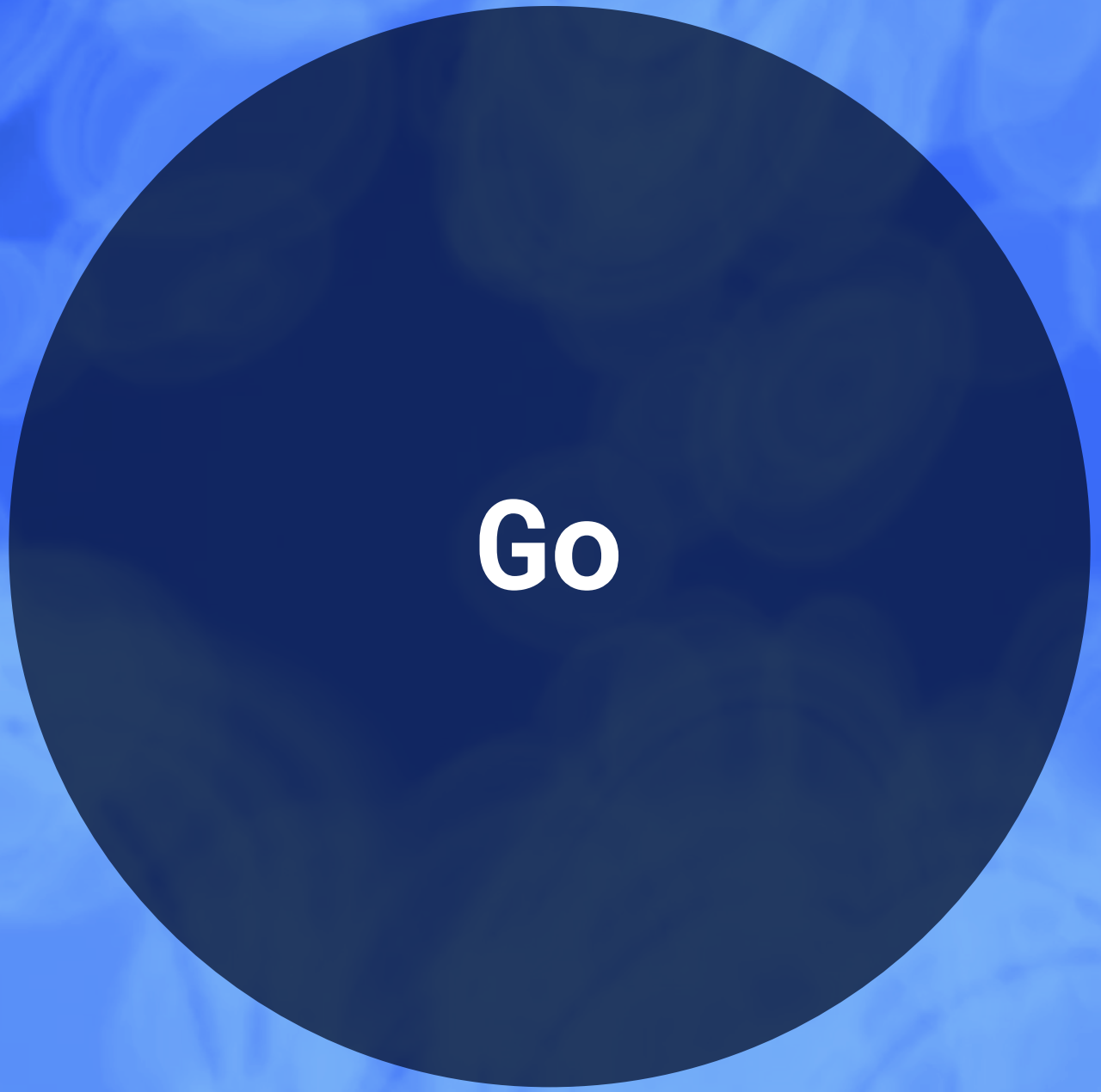
Symbiosis: Meta

> demo

https://github.com/AssemblyScript/assemblyscript

http://localhost:1234

https://github.com/maudnals/wasm-as


Go

> go

LEARN
= fundamentals

EXPERIMENT

USE
= practice

> go

LEARN
= fundamentals

EXPERIMENT

USE
= practice

MDN

webassembly
.studio

C tutos, AS
boilerplate,
rust-wasm
book

developers
.google.com

- Replacing a hot path in your JS app with Wasm
- Threads

libs

react-
wasm

...

TL;DL

🙏

Thank you

QnA time

> sources

Photos, Videos	https://www.pexels.com/
Illustrations, Diagrams	@maudnals
Part 1 – JS and Wasm	v8.dev/blog/liftoff mathiasbynens.be/notes/shapes-ics developer.mozilla.org/en-US/docs/WebAssembly d2jta7o2zej4pf.cloudfront.net/
Part 2, 3, 4 – Symbiosis + Demo	developers.google.com/web/updates/2019/02/hotpath-with-wasm webassembly.org/ github.com/AssemblyScript/assemblyscript
More	app.sketchup.com/app blogs.dropbox.com/tech/2018/06/building-better-compression-together-with-divans/ webassembly.org/docs/future-features/

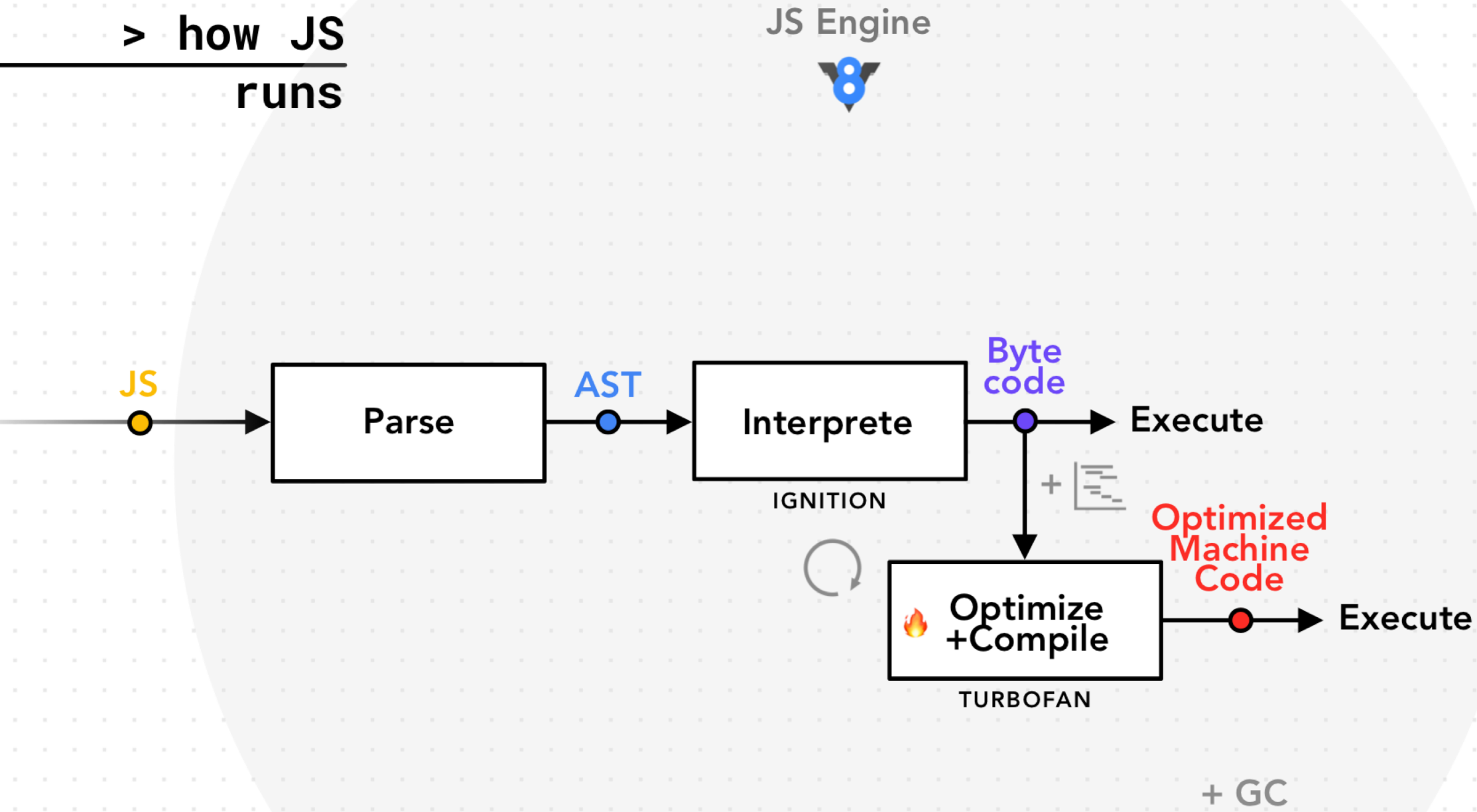
> resources

developers.google.com/web/updates/2019/02/hotpath-with-wasm
developers.google.com/web/updates/2018/03/emscripting-a-c-library
webassembly.org/
hacks.mozilla.org/
rustwasm.github.io/book/

github.com/AssemblyScript/assemblyscript

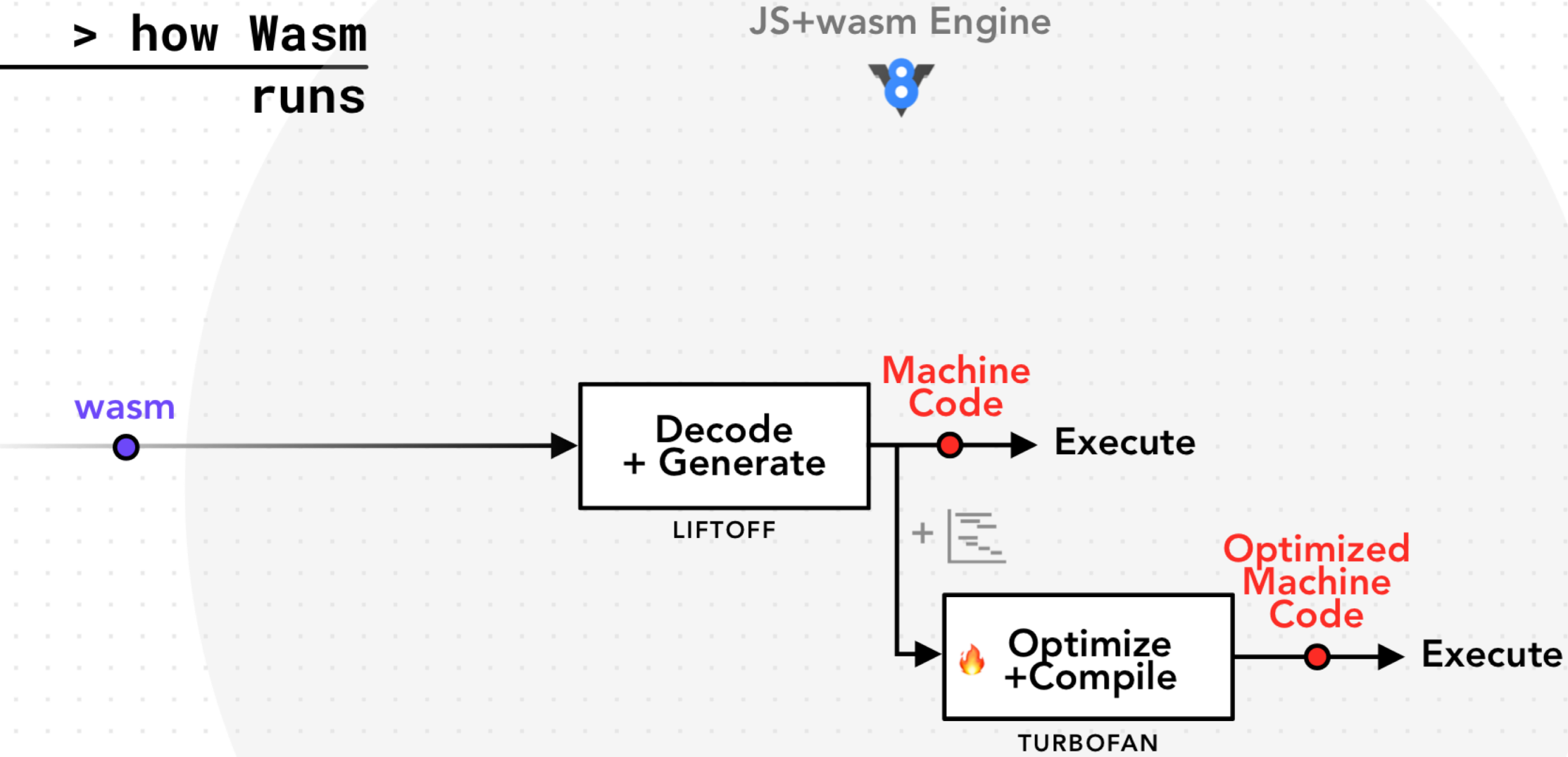
github.com/maudnals/wasm-as
github.com/maudnals/wasm-nano-handbook

> how JS runs

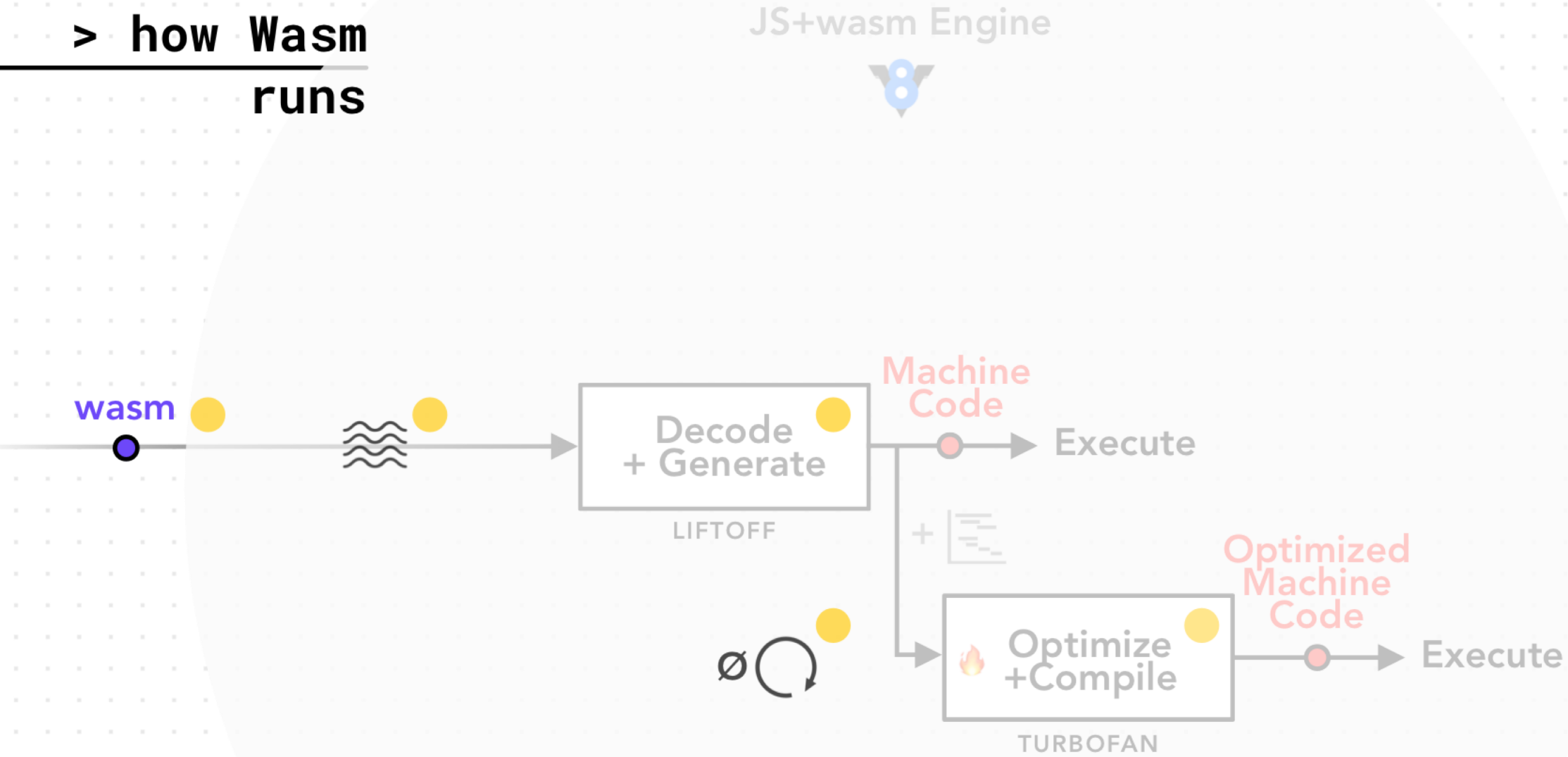


@maudnals

> how Wasm runs



> how Wasm runs



@maudnals