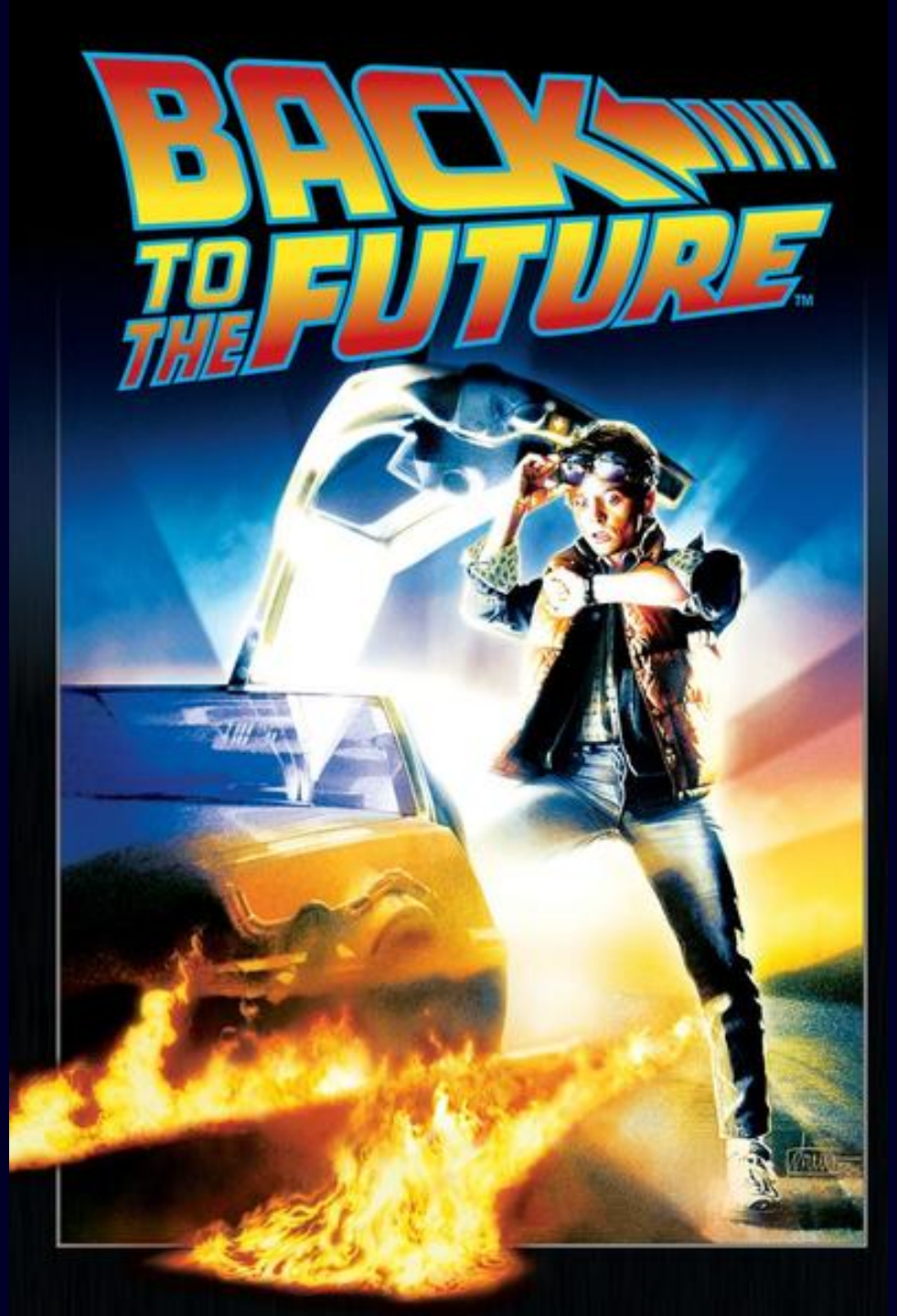
VIZBII

WebAssembly, Back to the Future?

Date : 04 / 06 / 2019



Sommaire

VIZBII

1

WebAssembly,
What is it?
Where does it come from?

VIZBII

50% Web

50% Assembly

Definition

WebAssembly or wasm is a new portable, size- and load-time-efficient format suitable for compilation to the web.

webassembly.org

1 Definition

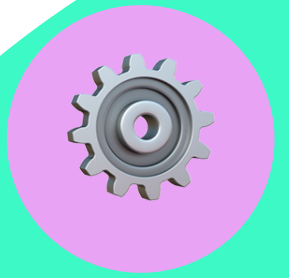
Key elements from this definition

FAST

At least better than javascript

PORTABLE

Should run on all (modern) browsers



COMPILED

Benefits from all the good things compiled languages can do

VIZBII

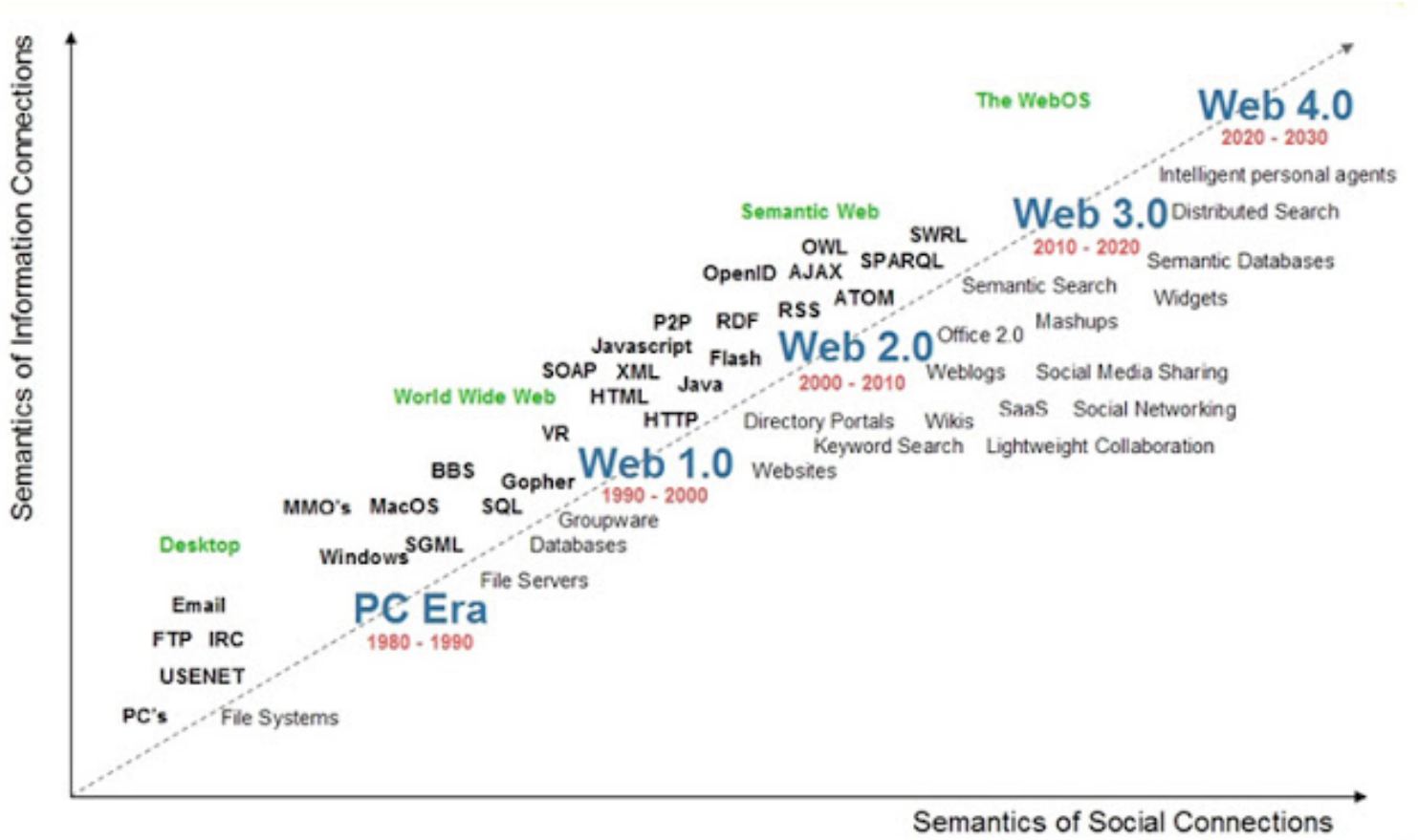
JS is truly a thing of wonder

```
> [] == ![]
< true
> NaN == NaN
< false
> Number.MIN_VALUE > 0
< true
> parseInt('fibii')
< NaN
> parseInt('fibii', 16)
< 15
> Math.min() > Math.max()
< true
```

1 Definition

—

The evolution of the web (2008)



Trying to get into the web

Here lies those who tried

non-exhaustive list

- Java
- Flash
- Silverlight
- ActiveX
- NaCl
- Dart

Why did they fail?

non-exhaustive list

- Compatibility issues
- Proprietary
- Safety

VIZBII

2

Conception

What do we want?

Human writable

- C/C++
- Rust
- Go
- ~~Brainfuck~~

+

Browser readable

- Easy to load
- Pre-optimized
- Bytecode

+

Machine executable

- Machine instructions
- System calls
- Safe

Compiling C to JS then?

VIZBII

ASM.js,
our yellow savior

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

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

2 Conception

—

ASM.js, the messiah

Project at Mozilla in 2013

Porting C code to a (performant) subset of javascript

Project was a success

Porting of Unreal Engine 3 took only 4 days

Emscripten’s LLVM backend new output : ASM.js

Laying out solid foundations for a portable, safe and fast language

How does this work?

0 Definition

—

From ASM.js to WebAssembly

Organization

- All major browsers representatives were invited
- W3C : WebAssembly Working Group

Goals

- Creating a stack-based VM
- Standardized
- Safe code
- Portable code
- Fast code

VIZBII

3

Inner workings

How does this work?

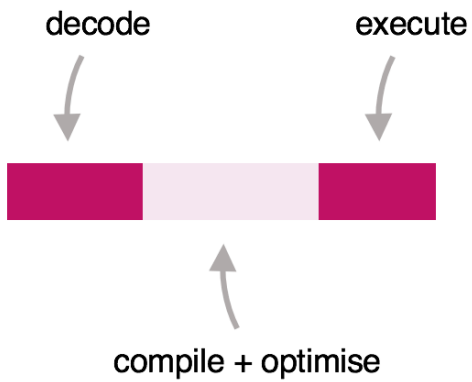
3 Inner workings

—

In the browser

JIT vs AOT

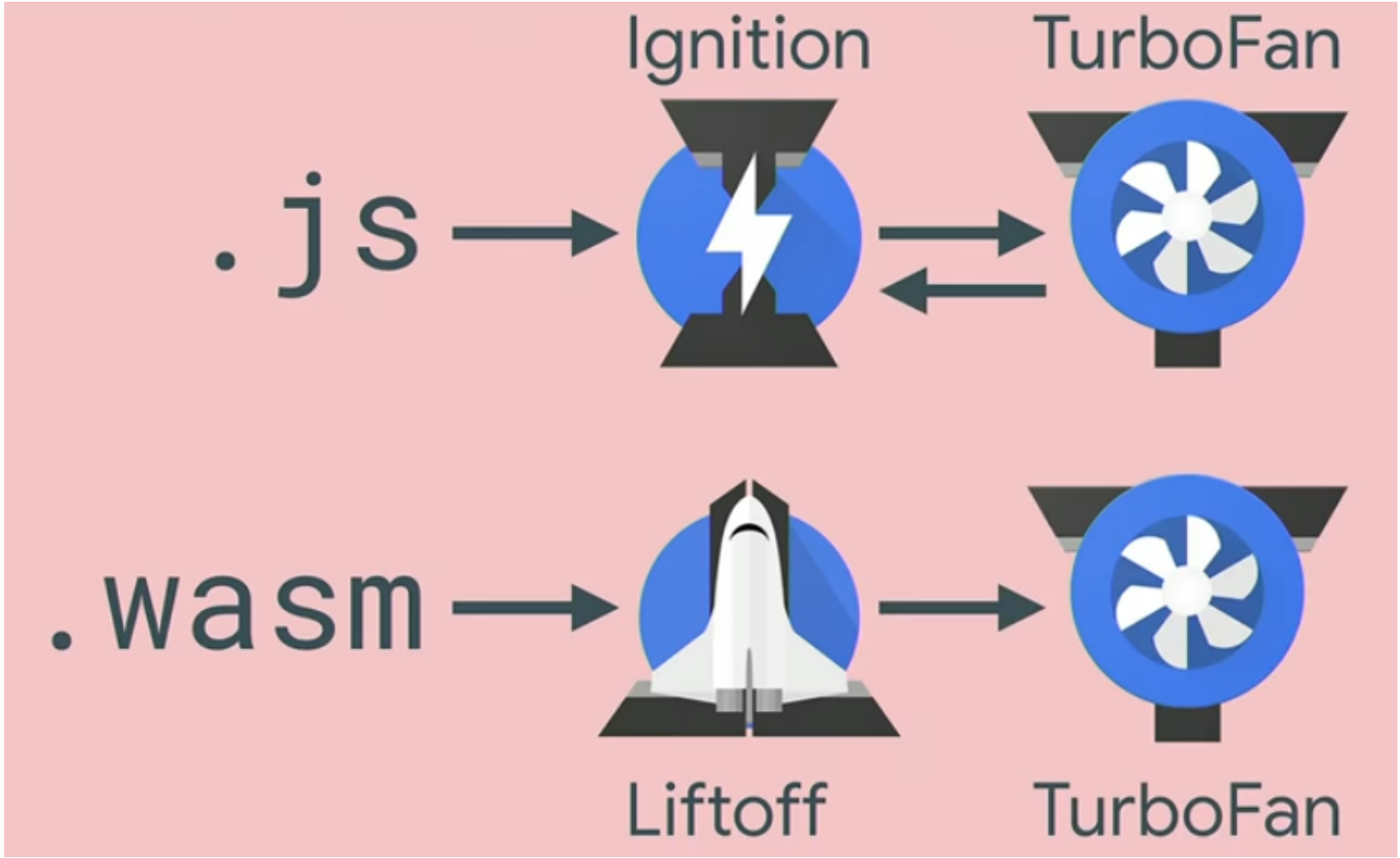
- JS is interpreted
- WebAssembly is compiled



3 Inner Workings

—

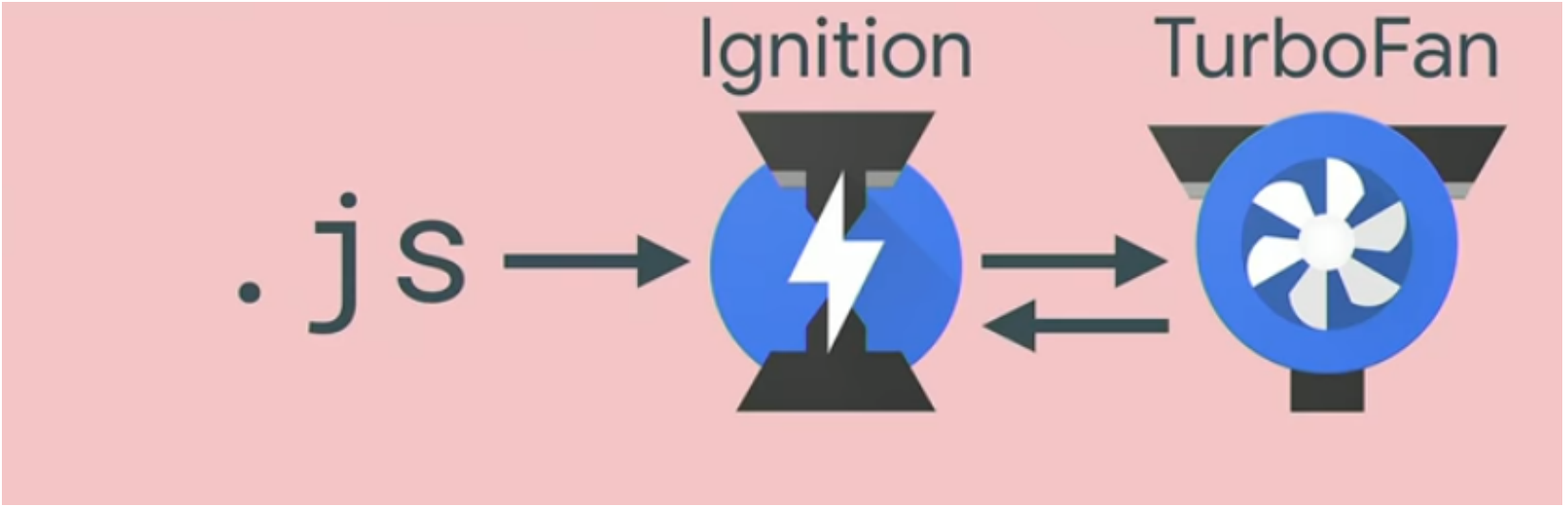
Chrome’s V8



3 Inner Workings

—

Chrome’s V8



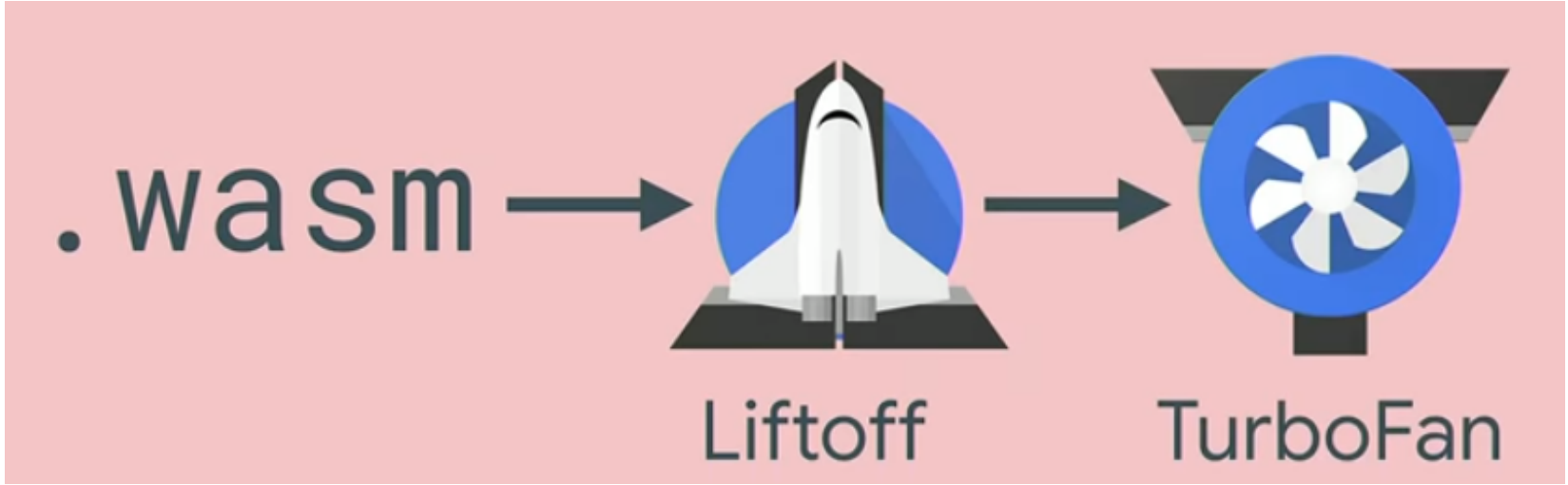
- Normal execution (JIT)
- A “hot-path” is found
- TF kicks in to run optimized JS
- TF might de-optimize
- Back to Igniter to interpret JS
- A “hot-path” is found
- …

3 Inner Workings

—

Chrome’s V8

- WASM is loaded
- WASM is already optimized
- Wasm is executed
- Done

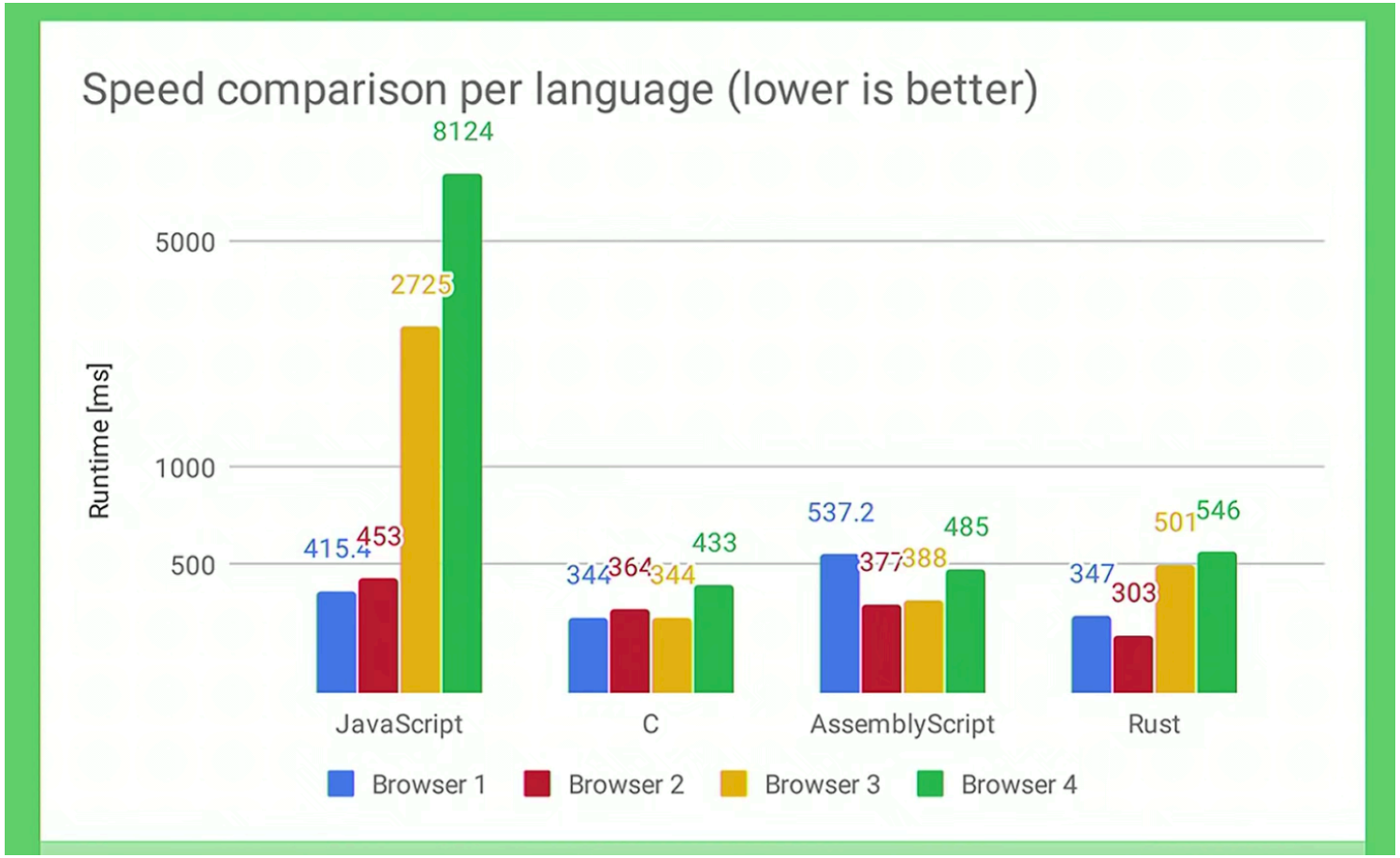


3 Inner Workings

—

Wasm Performance

- 10% slower than native C/C++
- Around 30% faster than JS
- Even closure compiled JS
- Low-end devices benefit from WASM even more!



What does it look like?

C WAT ASM

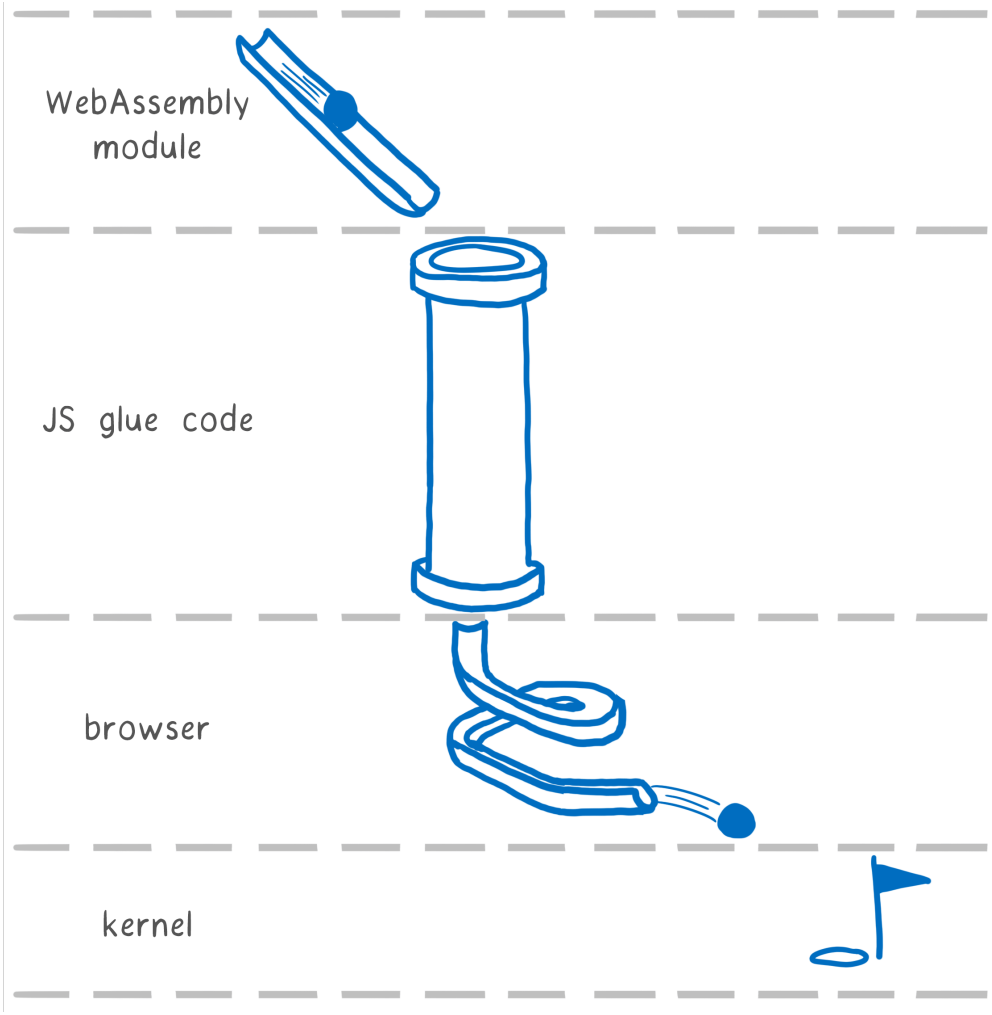
3 Inner workings

—

How do I call my WASM code?

Javascript I choose you!

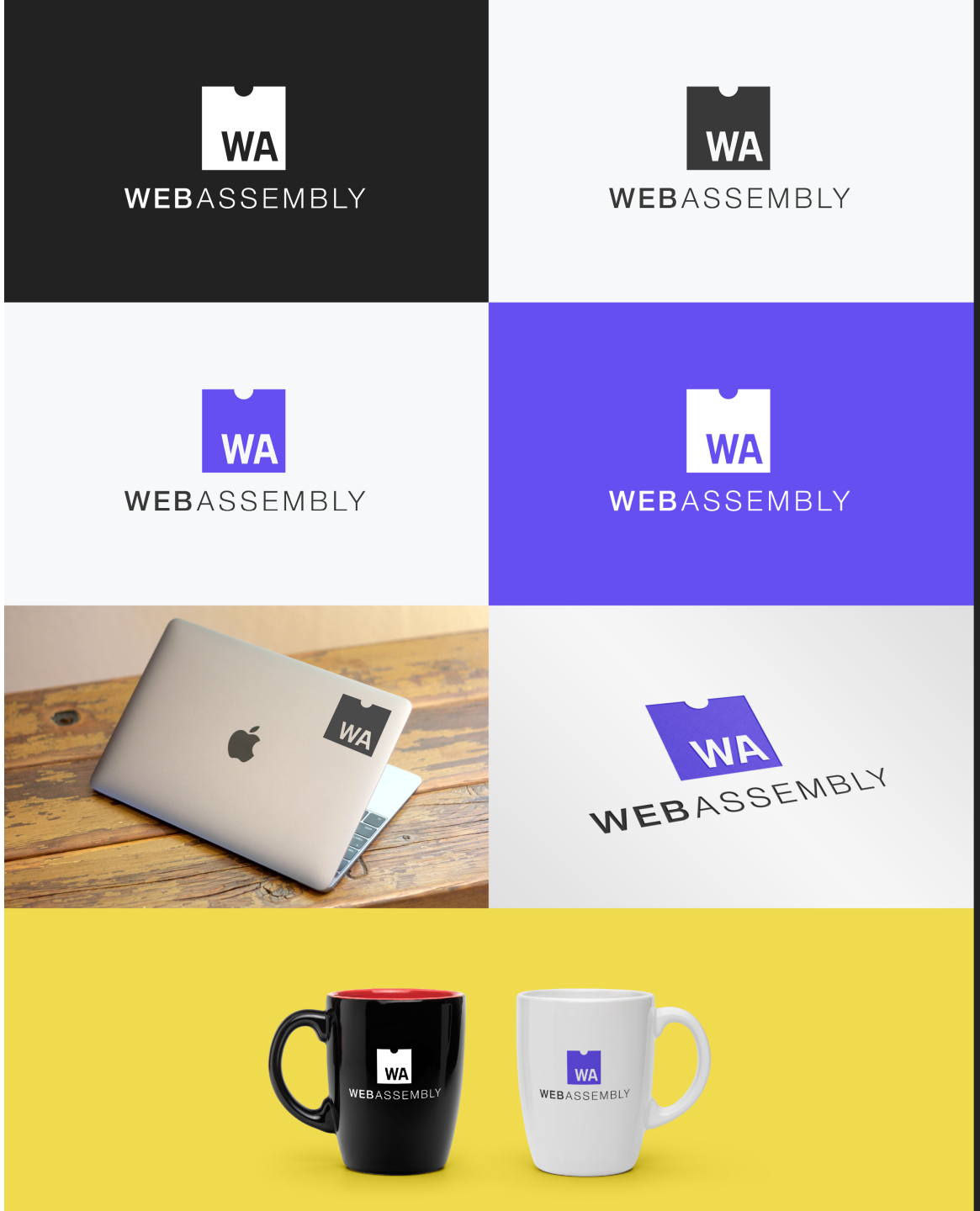
- Only JS is able to interact with the DOM
- JS will load WASM modules
- Some “glue” code might be required
- Safe way to interact with the system



VIZBII

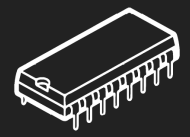
A strong dependency with JS

They’re best buddies



WASM LOGO INSPIRATION

1. **Integrated Circuit** *(IC)*



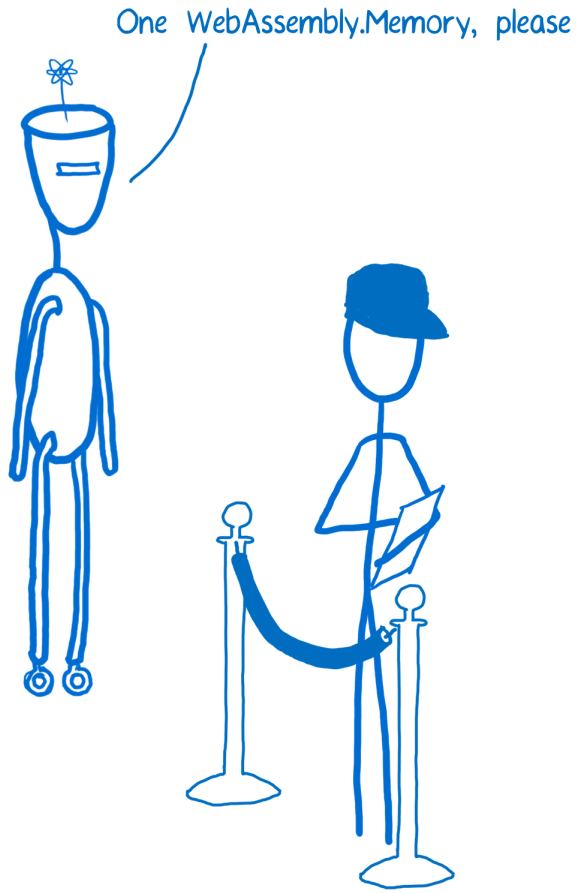
Top **notch** like the orientation notch of an IC

Low level: data structures, memory and CPU efficiency, etc

2. **WASM complements JS**

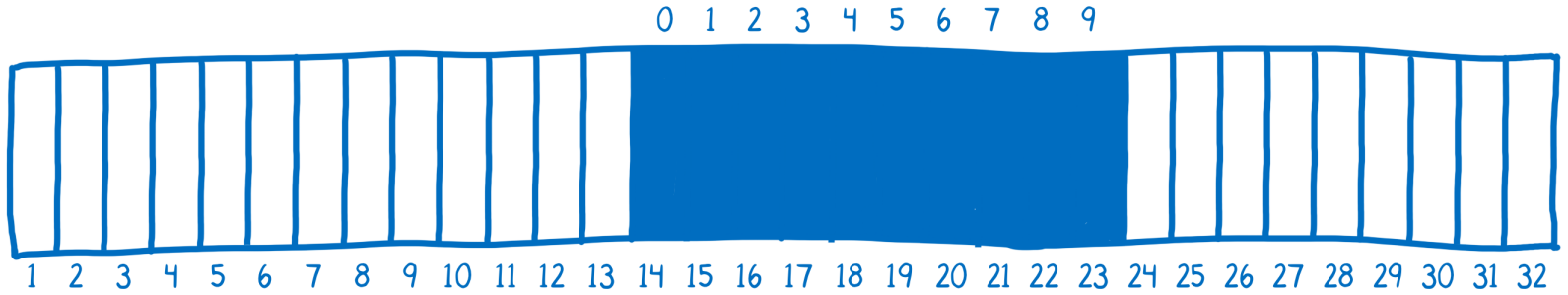
Similar **boxed logo** emphatises the relation between JS and WASM

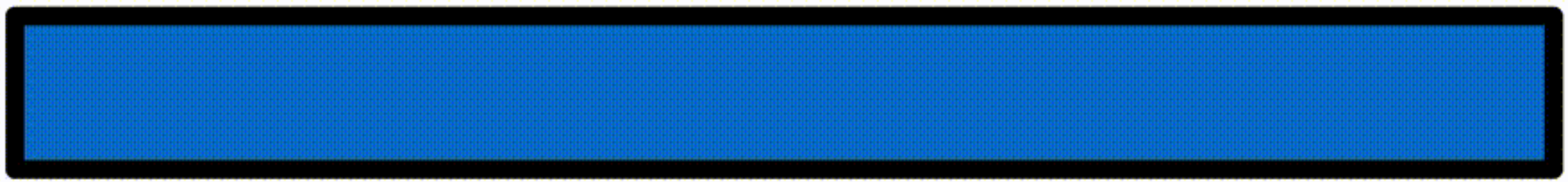
Primary color is JS complementary color



You shall only pass numbers between functions

- But there is a memory array
- Write JS object in
- Pass pointers to the data as numbers
- Read data from the buffer in WASM





WebAssembly.Memory

3 Inner workings

—

Passing pointers to data...

...nobody wants that

This is what the “glue” code might looks like.

Does anyone want to write this ?

```
import * as wasm from './foo_bg'; // imports from wasm file

const heap = new Array(32);
heap.push(undefined, null, true, false);
let heap_next = 36;

function addHeapObject(obj) {
  if (heap_next === heap.length)
    heap.push(heap.length + 1);
  const idx = heap_next;
  heap_next = heap[idx];
  heap[idx] = obj;
  return idx;
}

export function foo(arg0) {
  const idx0 = addHeapObject(arg0);
  wasm.foo(idx0);
}

export function __wbindgen_object_drop_ref(idx) {
  heap[idx ] = heap_next;
  heap_next = idx;
}
```

VIZBII

A little dependency with the ecosystem

```rust
#[wasm_bindgen]
extern "C" {
    fn alert(s: &str);
}

#[wasm_bindgen]
pub fn read_filename(filename: &str) {
    alert(&format!("Hello, {}!", filename));
}
```

Going full circle?

1 — Write in assembly

2 — Compile to assembly

3 — Static typing

4 — Interpreted language

5 — Javascript

6 — Typescript

7 — WebAssembly

8 — Write in assembly ?

VIZBII

4

Use cases

4 Use cases

—

Autodesk

Autocad

-35 years old codebase

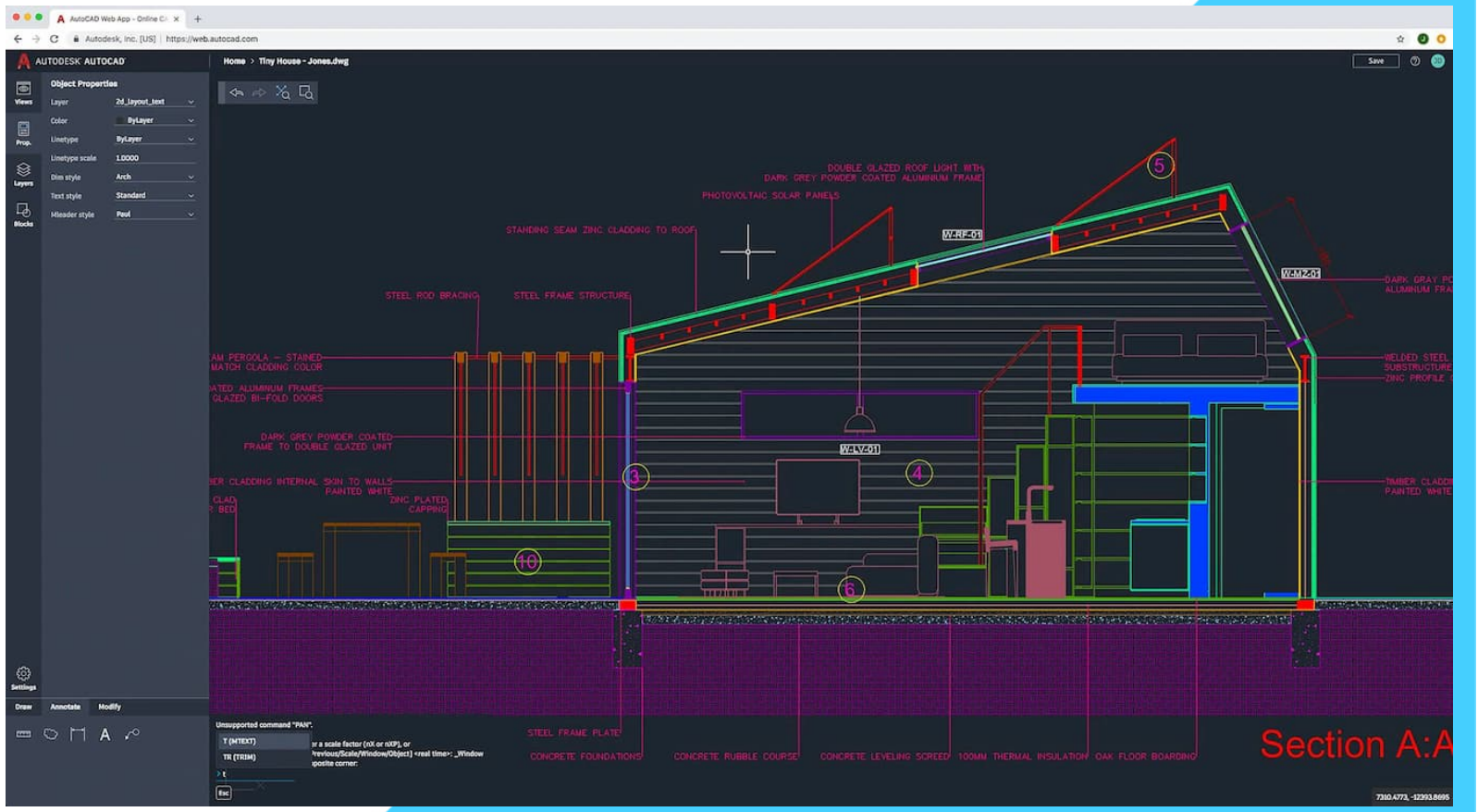
-Compile to WebAssembly

-JS bindings

-Read-only first

-Progressively porting existing c++ code

-Now run 100% in the browser



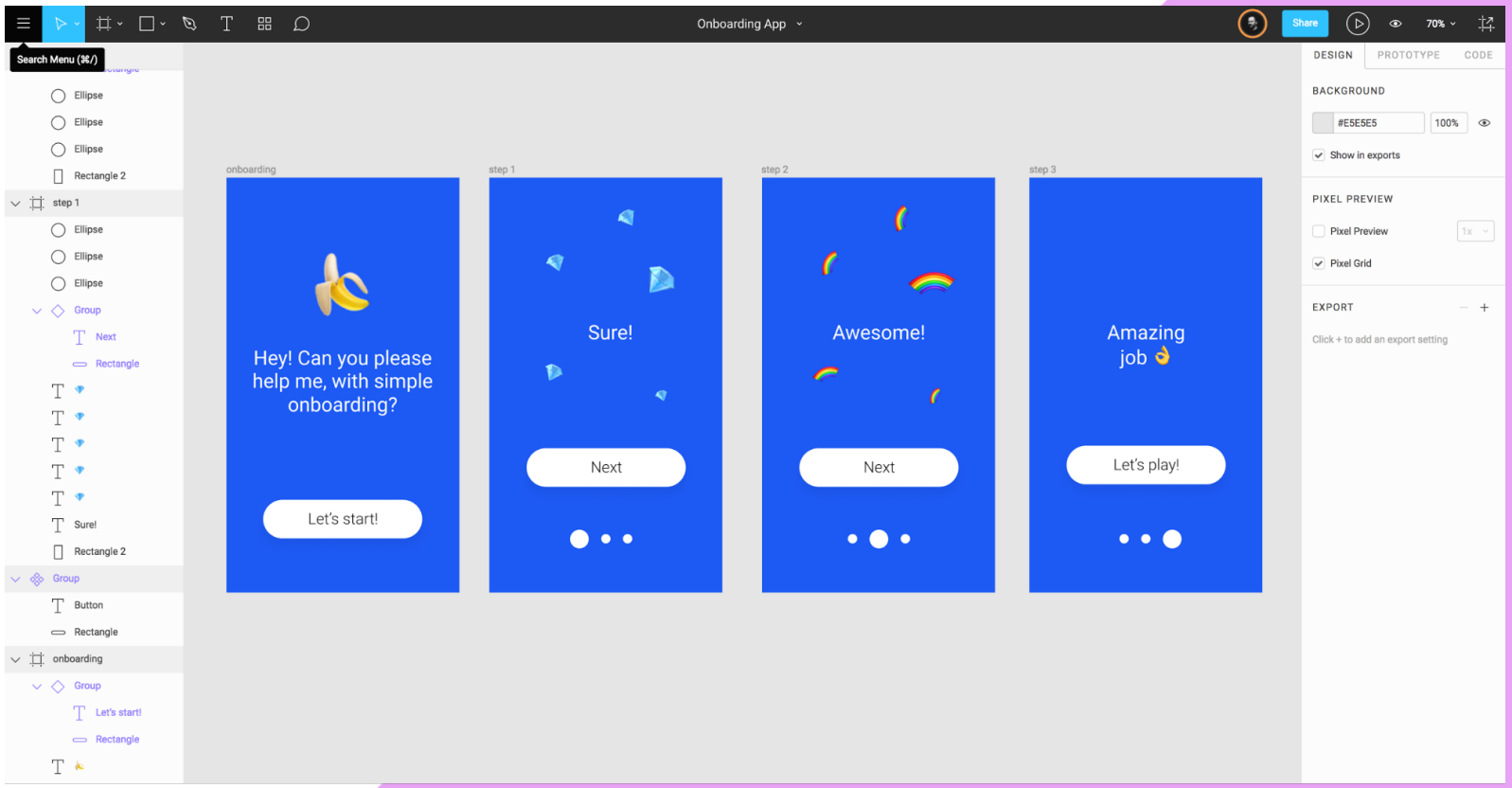
4 Use cases

—

Figma

Design tool

- Product initially written in C++
- Build with ASM.js
- WebAssembly = drop-in replacement
- Parsing is 20x faster than ASM.js
- Already optimized by the compiler
- 3x load time improvement



4 Use cases

—

Ebay

Barcode scanner

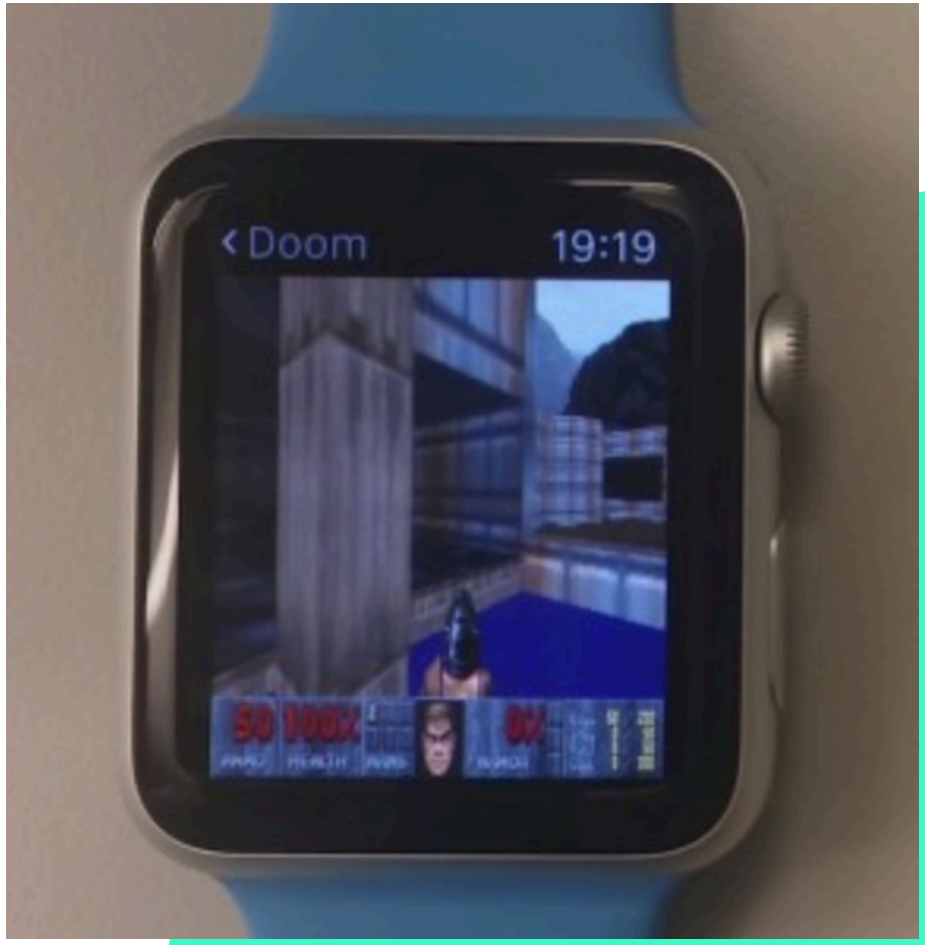
- C++ scanner code for native app
- JS version had poor performance
- WebAssembly to the rescue !

4 Use cases

—

And so much more...

- Game engines
- Video
- Image
- Sound
- ...



Some more examples

Game engine:
DOOM 3

- 7 weeks
- 40-60% slower than native
- 850MB in RAM

Image manipulation:
Squoosh.app

- Convertion
- Crop
- Resize
- Rotate

VIZBII

5

DEMO

1

Write in assembly

2

Compile to assembly

3

Static typing

4

Interpreted language

5

Javascript

6

Typescript

7

WebAssembly

8

Write in assembly ?

1

—

Write in assembly

2

—

Compile to assembly

3

—

Static typing

4

—

Interpreted language

5

—

Javascript

6

—

Typescript

7

—

WebAssembly

8

—

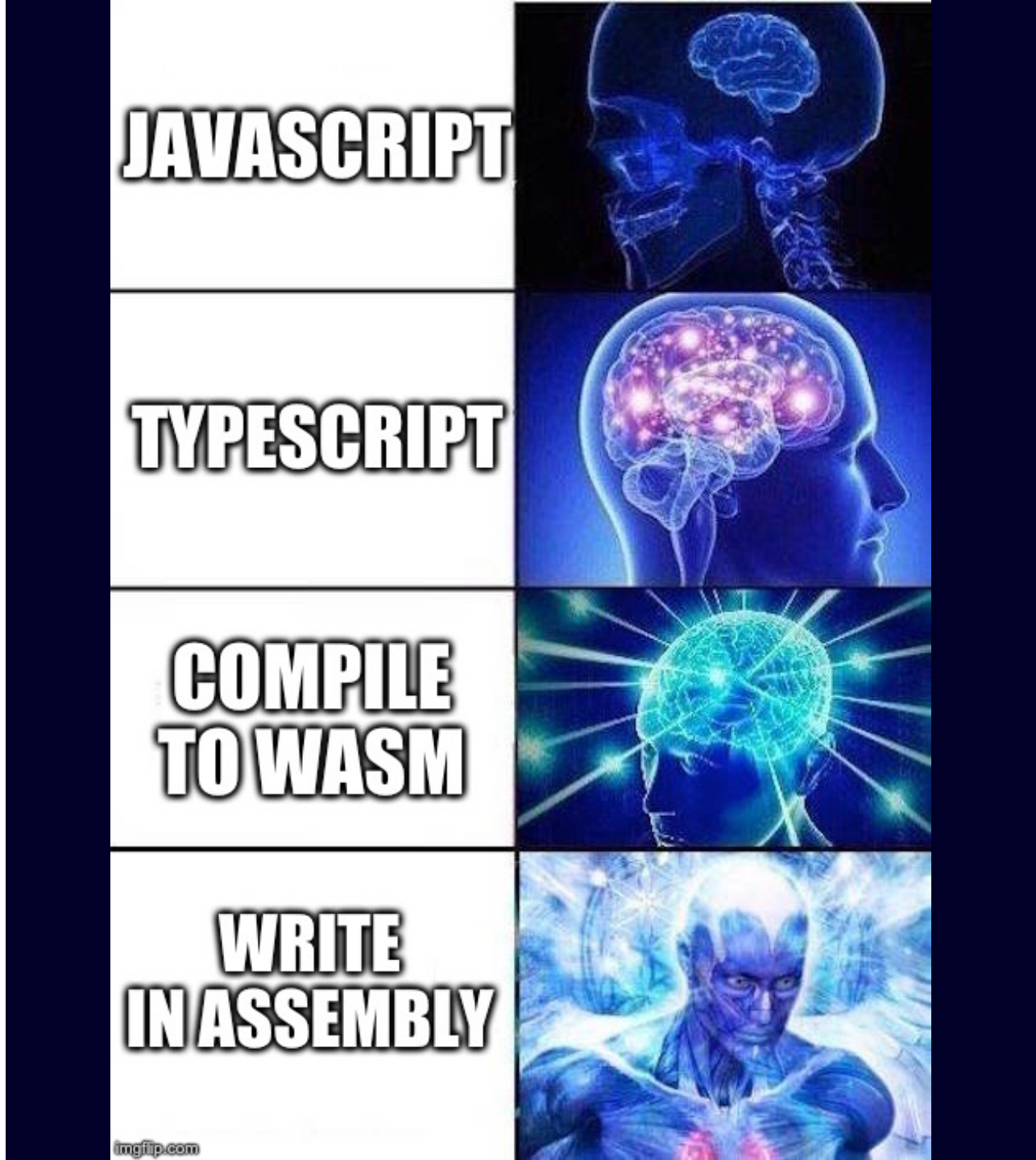
Write in assembly ?

VIZBII

Let’s start with the easy one

Writing in assembly for the browser

Yes we’re really gonna do that, I’m so sorry



4 Demo

—

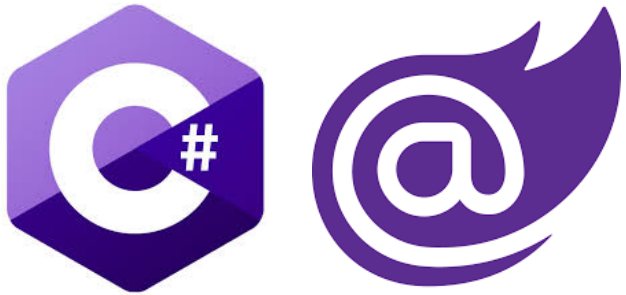
What if I need to convince my CTO?

Hello World

C++ / emscripten

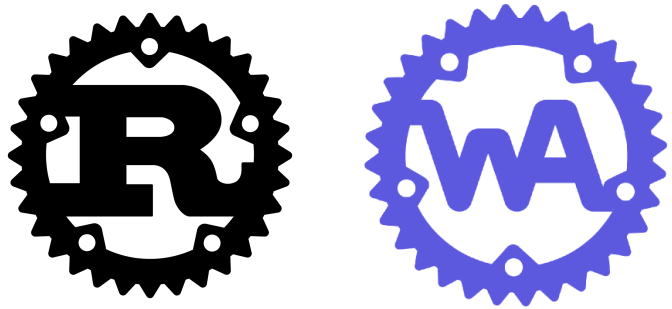
Web App

C# / blazor



Leveraging libs

Rust / wasm-bindgen



VIZBII

Rust ecosytem is reeeeeealy good 🦀

WASM is on the ~~(M)~~Loose

Definition

WebAssembly or wasm is a new portable, size- and load-time-efficient format suitable for compilation to the web.

webassembly.org

Definition

WebAssembly is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable target [...] for client and server applications.

webassembly.org

5 Epilogue

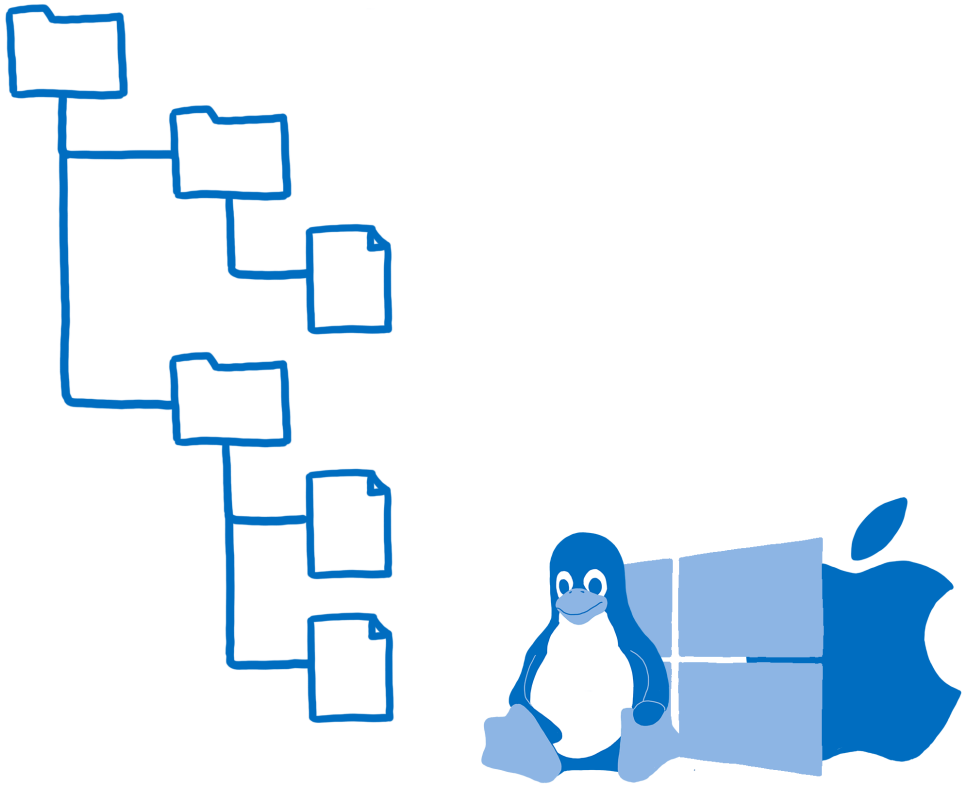
—

WASI, WebAssembly system interface

Running WebAssembly outside the web

- WASM runtimes
- Standardized system calls
- Safe execution everywhere
- WASM can now run outside a browser
- Already multiple runtimes available

5 Epilogue


Here you go. Here are some safe toys to interact with the OS.
Aw, man... Where is my network access!?
> risky-app
WA
safe_write
afe_read

VIZBII

One more thing...

1 — Threads

2 — Exceptions

3 — Garbage collection

Code: https://github.com/thuchede/wasm-intro

—

Look what we've built today!

WAT

Add function in a simple module

C++

Hello world built with emscripten

Rust

Simple App leveraging existing libs with wasm-bindgen

C#

Demo app : https://dotnet.microsoft.com/apps/aspnet/web-apps/client

References

—

https://webassembly.org/
https://github.com/guyroyse/intro-to-webassembly
https://slides.com/nnodot/mixit-2019-wasm
https://www.youtube.com/watch?v=njt-Qzw0mVY : WebAssembly for Web Developers (Google I/O '19)
https://blog.scottlogic.com/2019/05/17/webassembly-compiler.html
https://v8.dev/blog
https://www.figma.com/blog/webassembly-cut-figmas-load-time-by-3x/
https://www.ebayinc.com/stories/blogs/tech/webassembly-at-ebay-a-real-world-use-case/
http://www.continuation-labs.com/projects/d3wasm/
https://squoosh.app/

https://emscripten.org/
https://rustwasm.github.io/docs/book/
https://dotnet.microsoft.com/apps/aspnet/web-apps/client

https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/

https://imgflip.com/memegenerator/

https://github.com/mbasso/awesome-wasm/blob/master/README.md

VIZBII

Question(s)?

👋