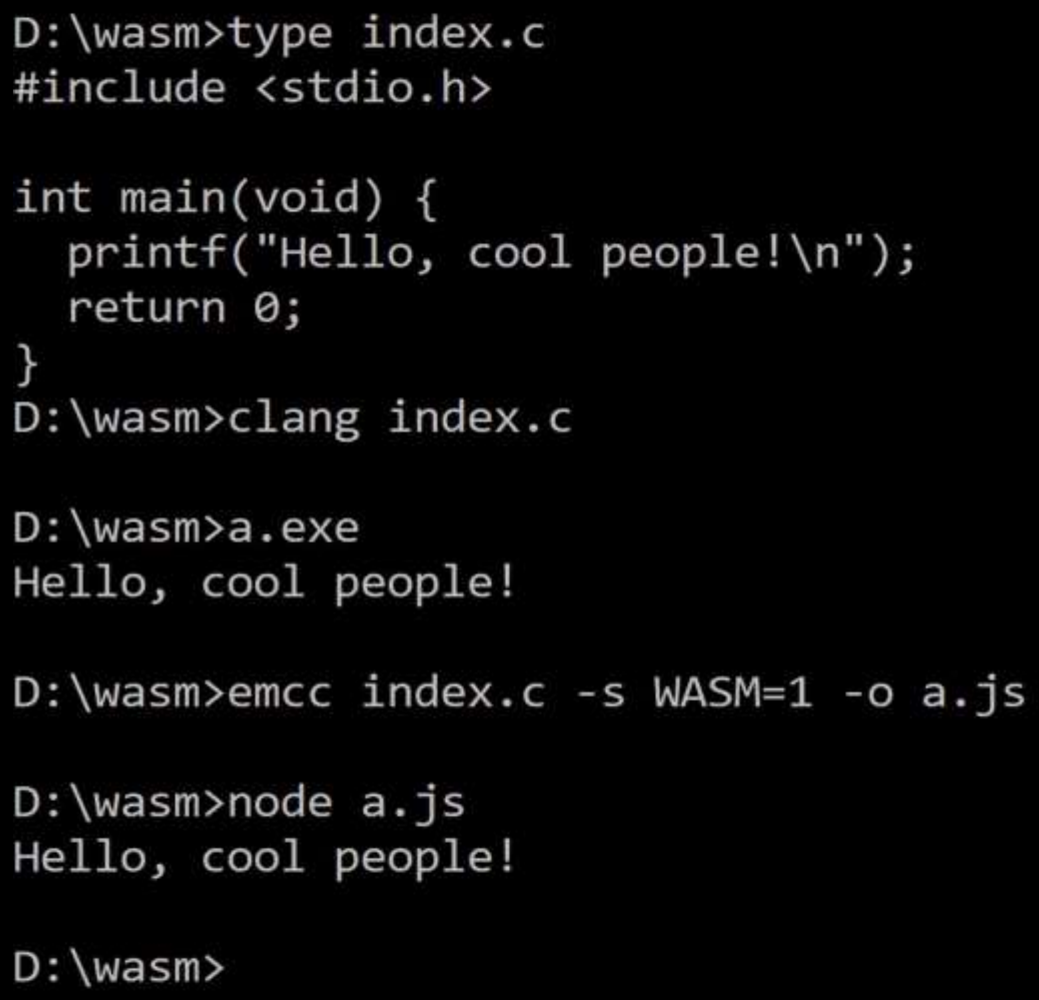
How WebAssembly is changing the Web and what it means for you

Boyan Mihaylov
@boyanio
boyan.io

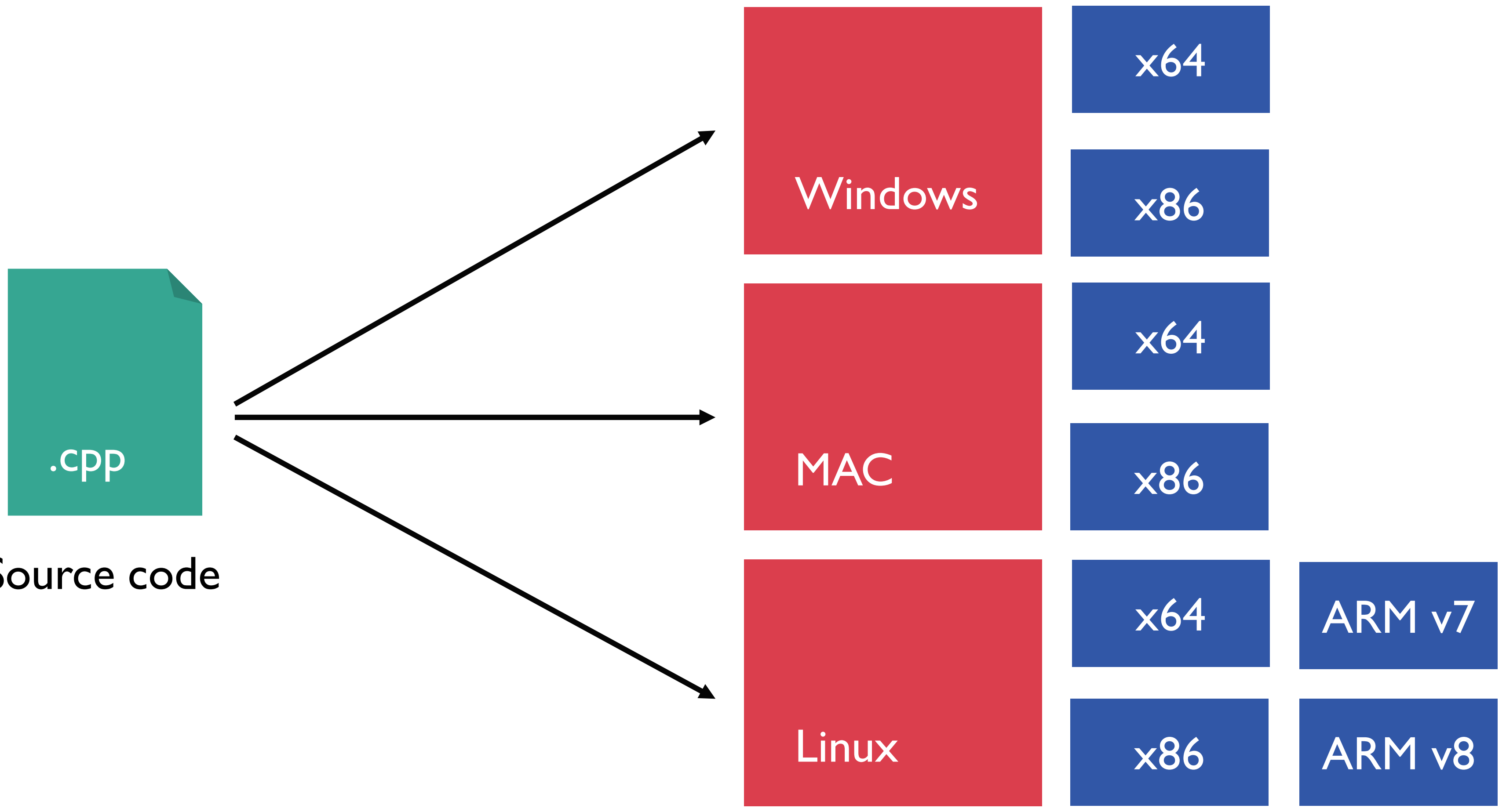
WebAssembly (*WASM*) is compiler target for programs on the Web

@boyanio

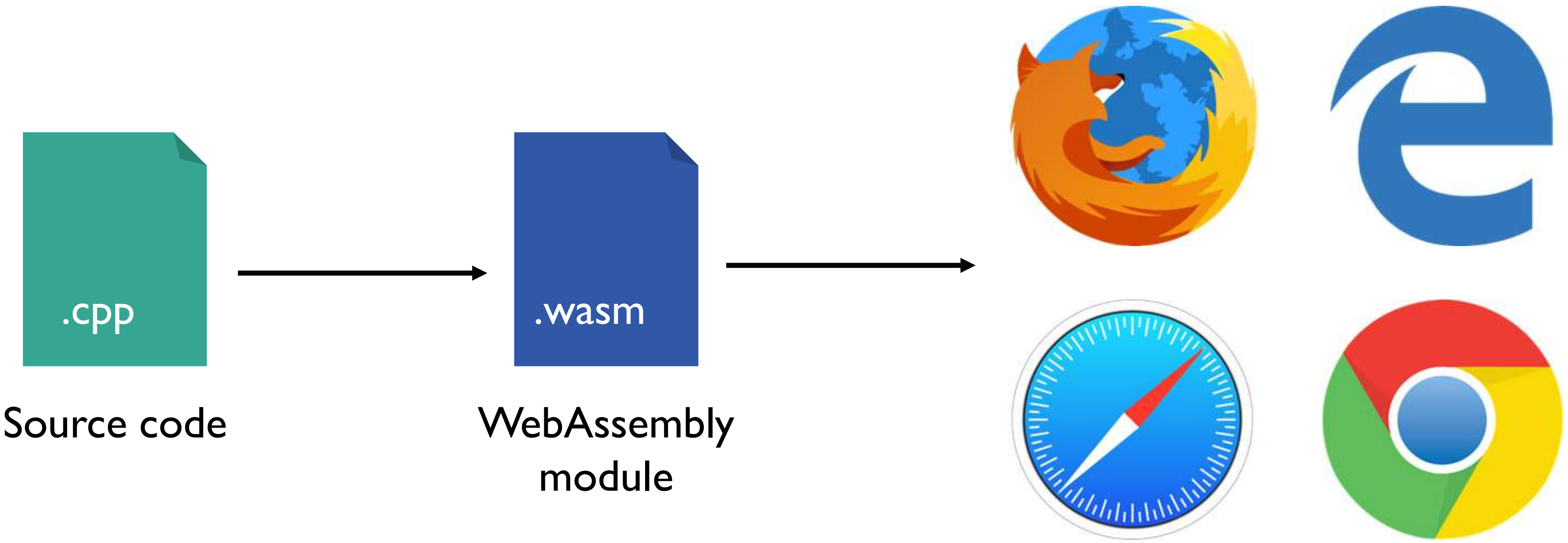


```
D:\wasm>type index.c
#include <stdio.h>

int main(void) {
  printf("Hello, cool people!\n");
  return 0;
}
D:\wasm>clang index.c

D:\wasm>a.exe
Hello, cool people!

D:\wasm>emcc index.c -s WASM=1 -o a.js

D:\wasm>node a.js
Hello, cool people!

D:\wasm>
```

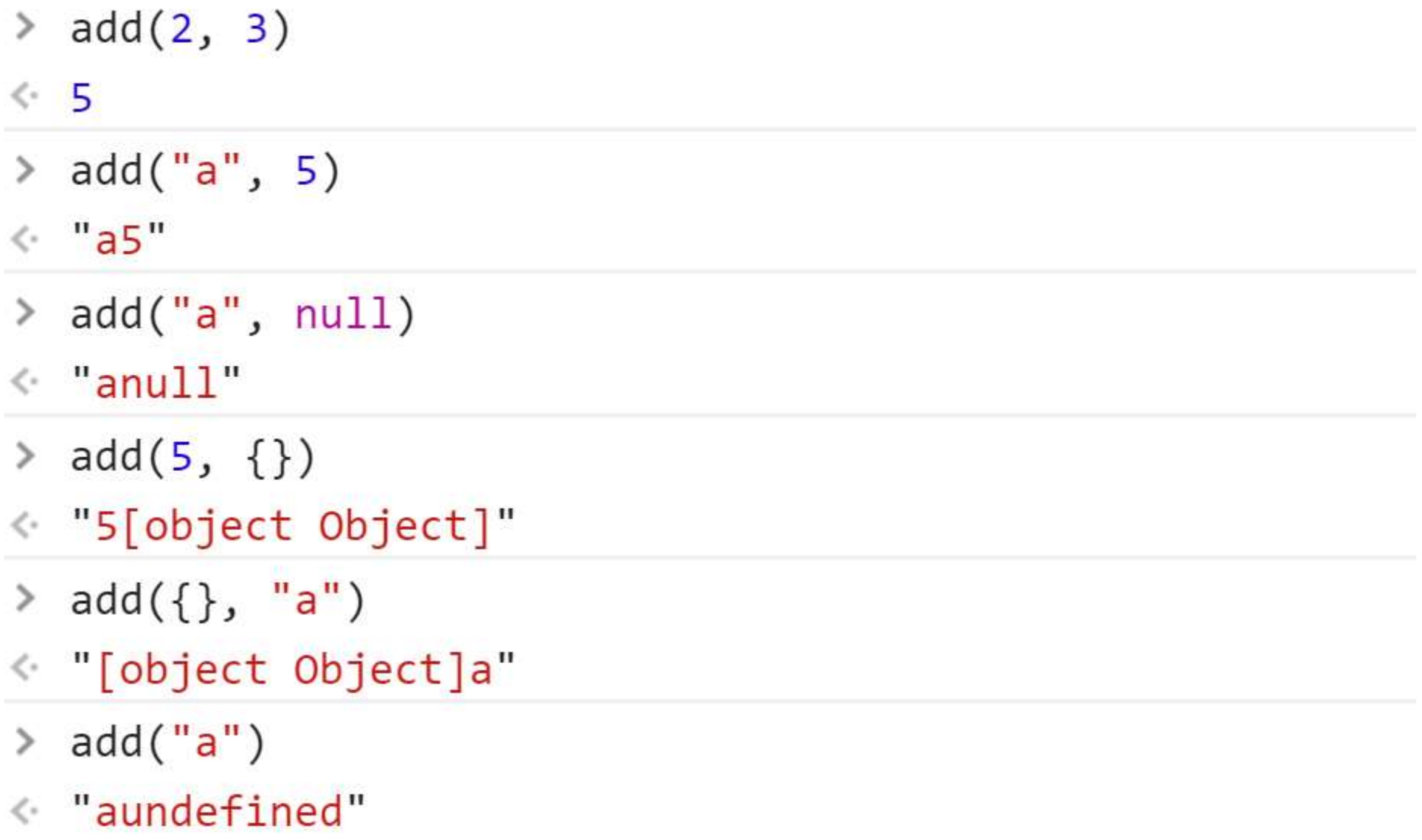
Traditional multi-target compilation



Multi-target compilation with WebAssembly

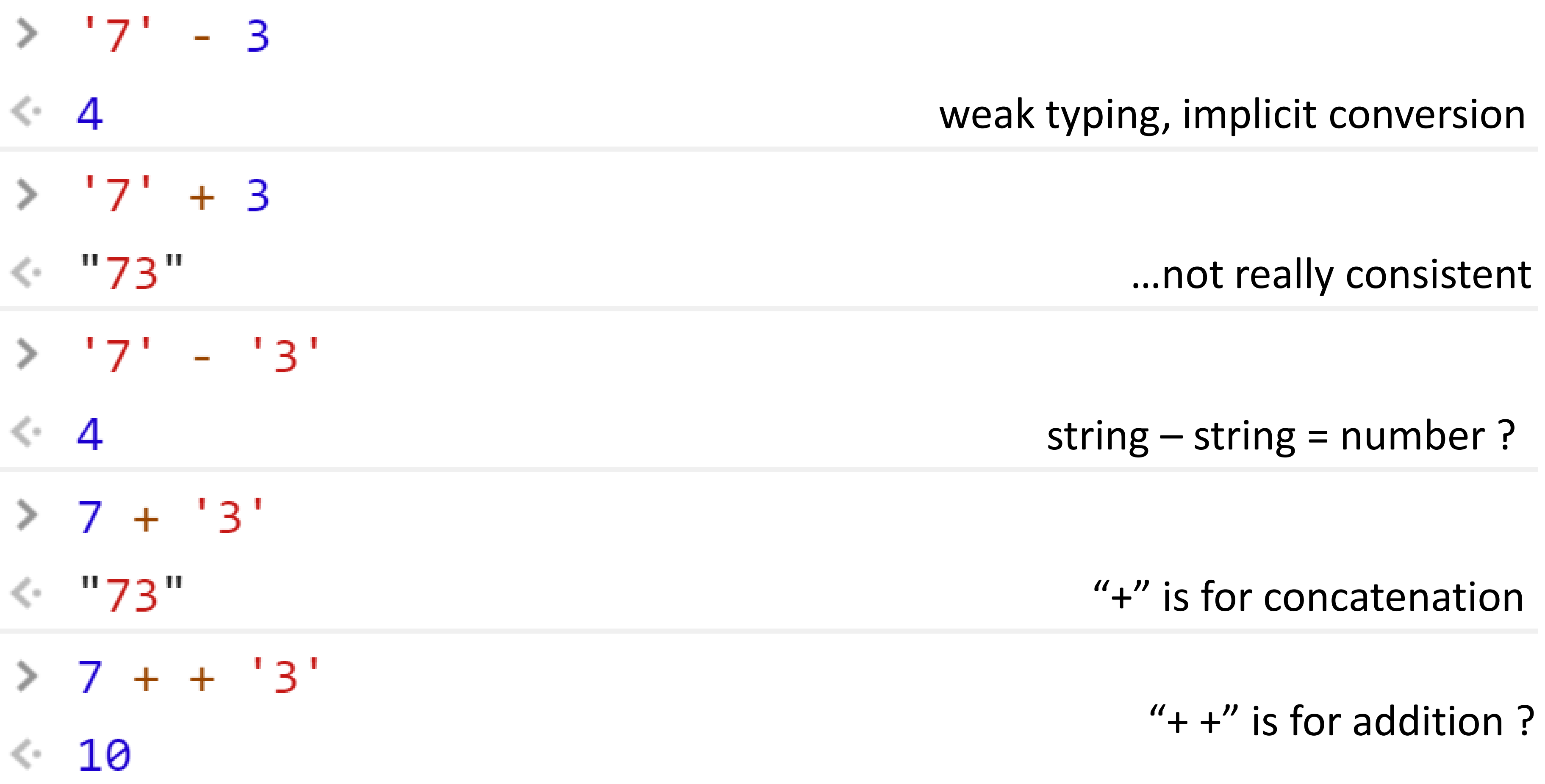


```
function add(a, b) {
  return a + b;
}
```



```
> add(2, 3)
<· 5
> add("a", 5)
<· "a5"
> add("a", null)
<· "anull"
> add(5, {})
<· "5[object Object]"
> add({}, "a")
<· "[object Object]a"
> add("a")
<· "aundefined"
```

@boyanio



```
> '7' - 3
<· 4
```

weak typing, implicit conversion

```
> '7' + 3
<· "73"
```

…not really consistent

```
> '7' - '3'
<· 4
```

string – string = number ?

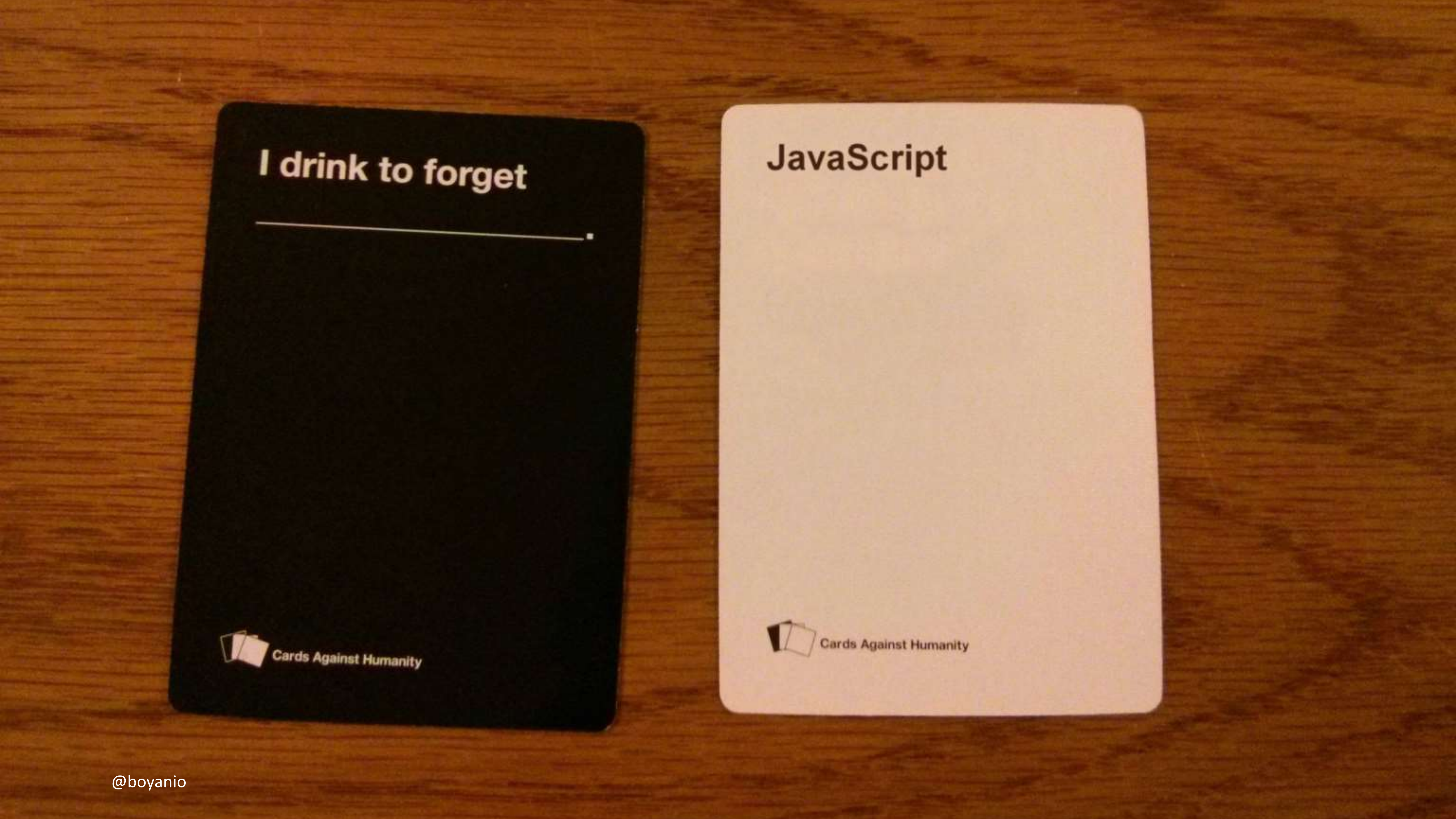
```
> 7 + '3'
<· "73"
```

"+" is for concatenation

```
> 7 + + '3'
<· 10
```

"+ +" is for addition ?

@boyanio


I drink to forget ________.
Cards Against Humanity
JavaScript
Cards Against Humanity

@boyanio

WebAssembly is a typed language

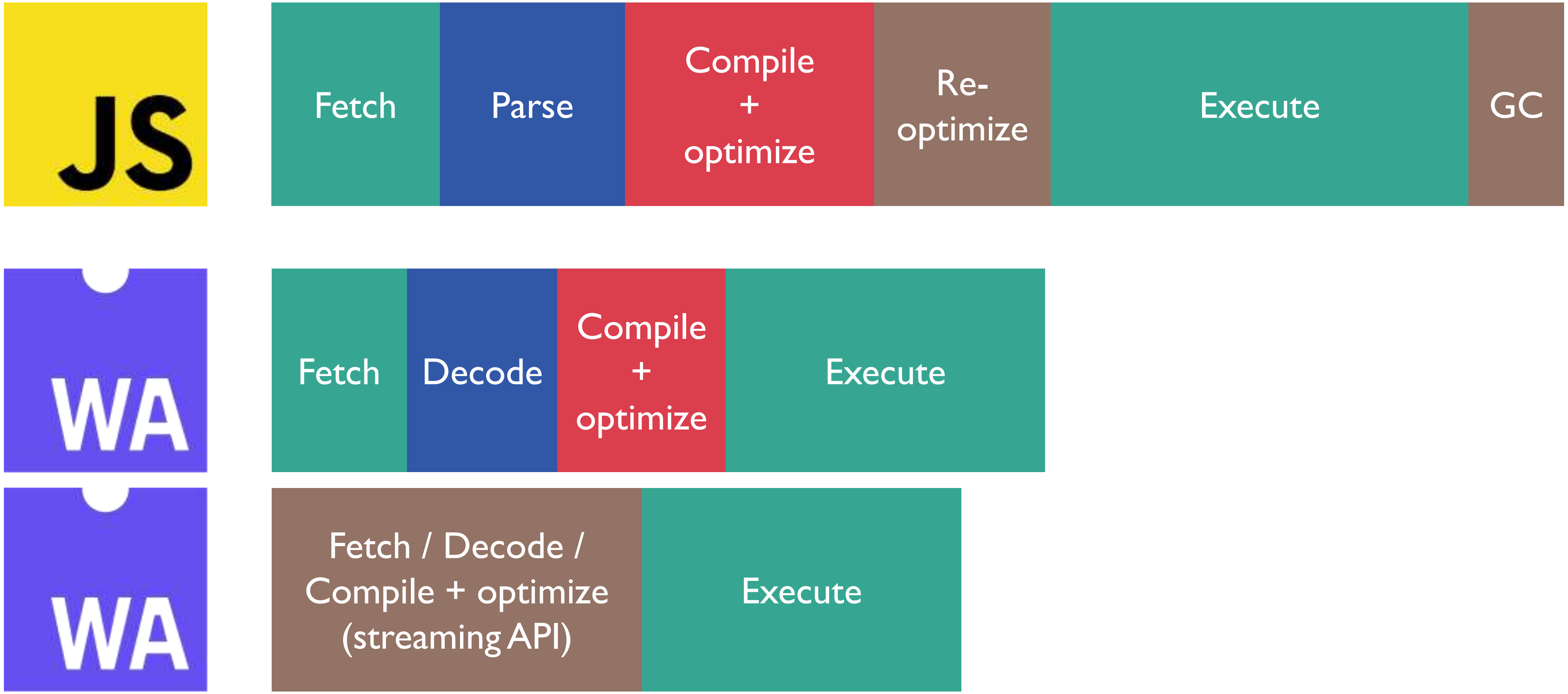
It supports 32 and 64-bit integers (i32, i64) and floating points (f32, f64)

Binary representation (.wasm)

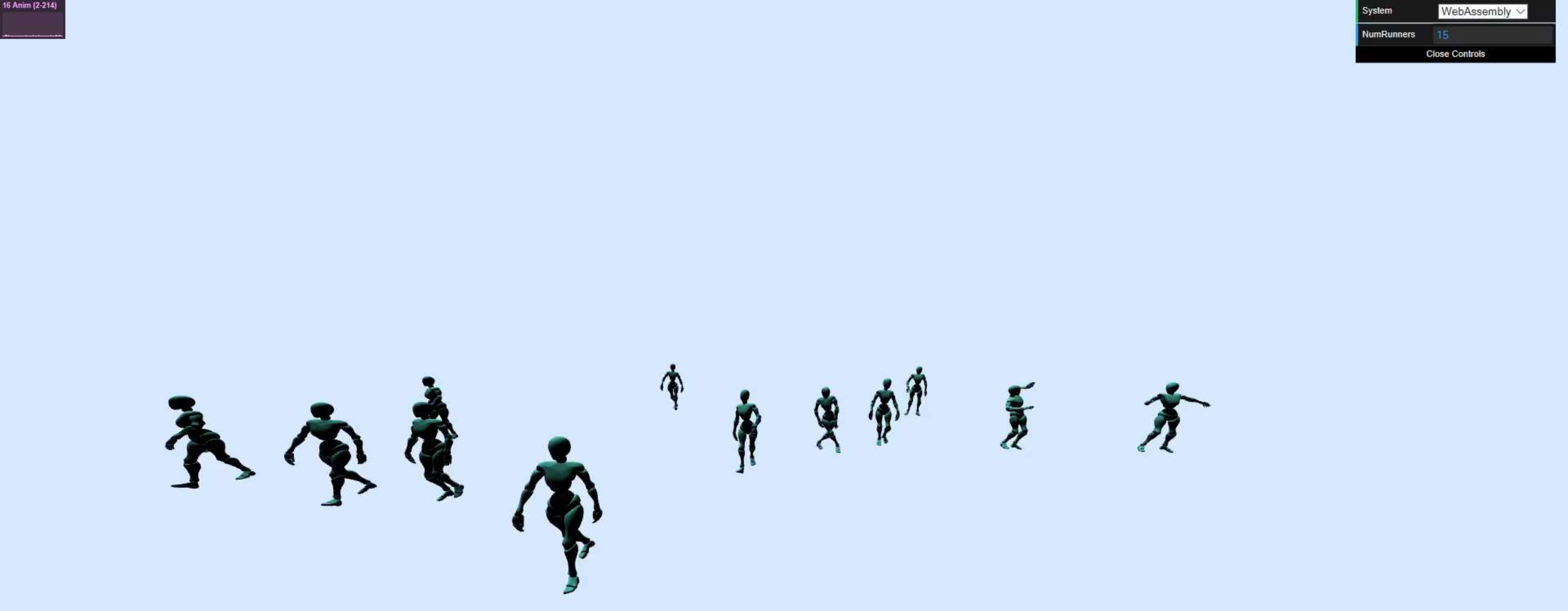
```
0061 736d 0100 0000 0187 8080 8000 0160
027f 7f01 7f03 8280 8080 0001 0004 8480
8080 0001 7000 0005 8380 8080 0001 0001
0681 8080 8000 0007 9080 8080 0002 066d
656d 6f72 7902 0003 6164 6400 000a 8d80
8080 0001 8780 8080 0000 2001 2000 6a0b
```

Textual representation (.wat)

```
(module
  (table 0 anyfunc)
  (memory $0 1)
  (export “memory” (memory $0))
  (export “add” (func $add))
  (func $add (; 0 ;) (param $0 i32) (param $1 i32) (result i32)
    (i32.add
      (get_local $1)
      (get_local $0)
    )
  )
)
```


JS
Fetch
Parse
Compile + optimize
Re-optimize
Execute
GC
WA
Fetch
Decode
Compile + optimize
Execute
WA
Fetch / Decode / Compile + optimize (streaming API)
Execute

WebAssembly provides consistent, predictable performance



System: WebAssembly

NumRunners: 15

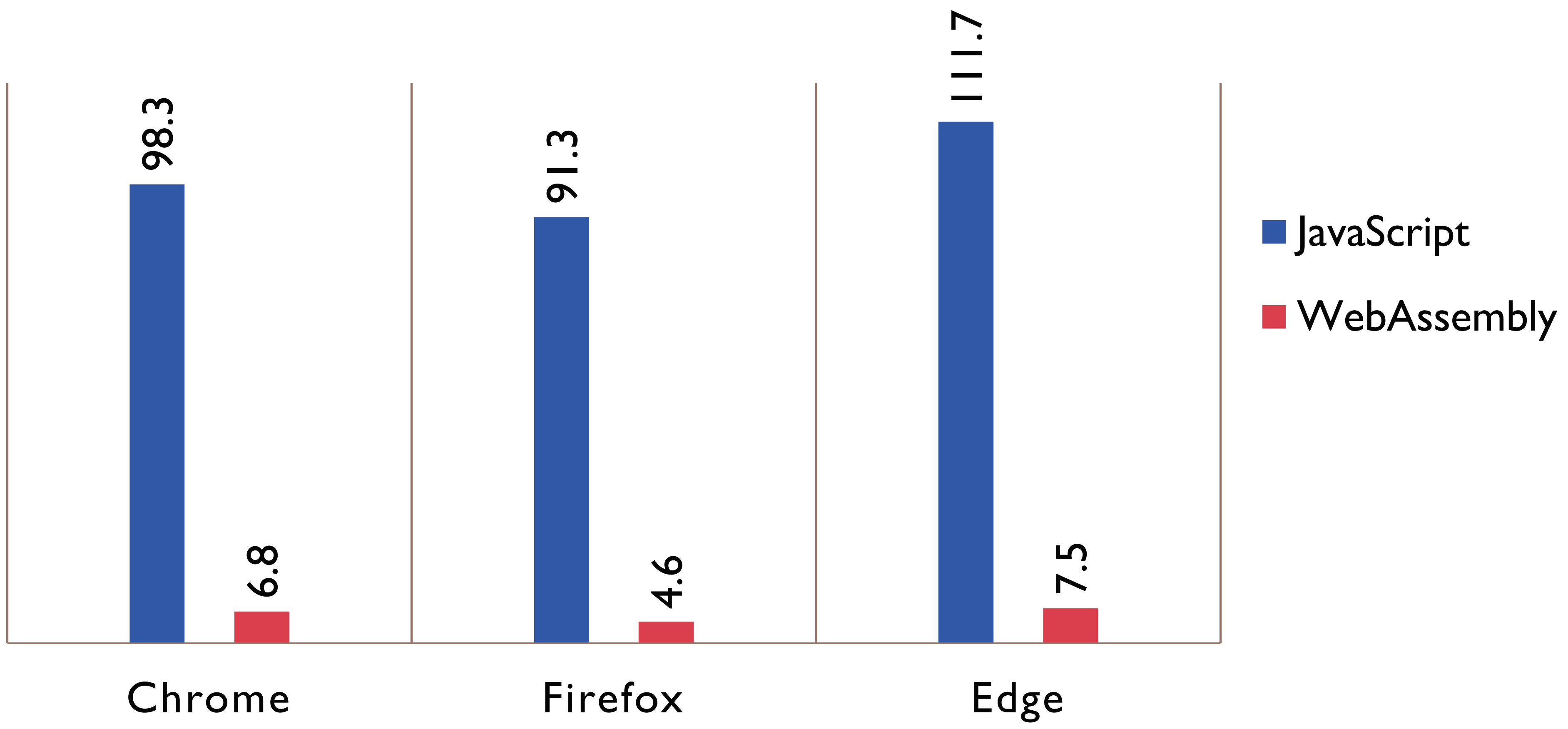
3D animation performance

https://github.com/sessamekesh/wasm-3d-animation-demo

@boyanio

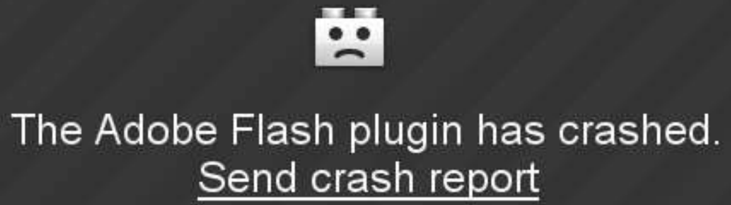
Performance comparison

Average animation time (ms)



@boyanio

https://www.lucidchart.com/techblog/2017/05/16/webassembly-overview-so-fast-so-fun-sorta-difficult/


The Adobe Flash plugin has crashed.
Send crash report

Reusing code on the Web

@boyanio

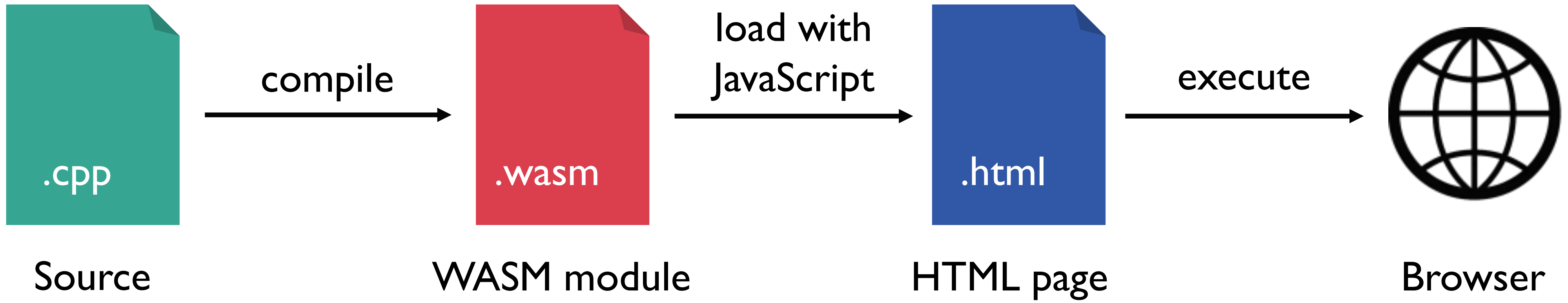
WebAssembly enables code reusability between native and Web

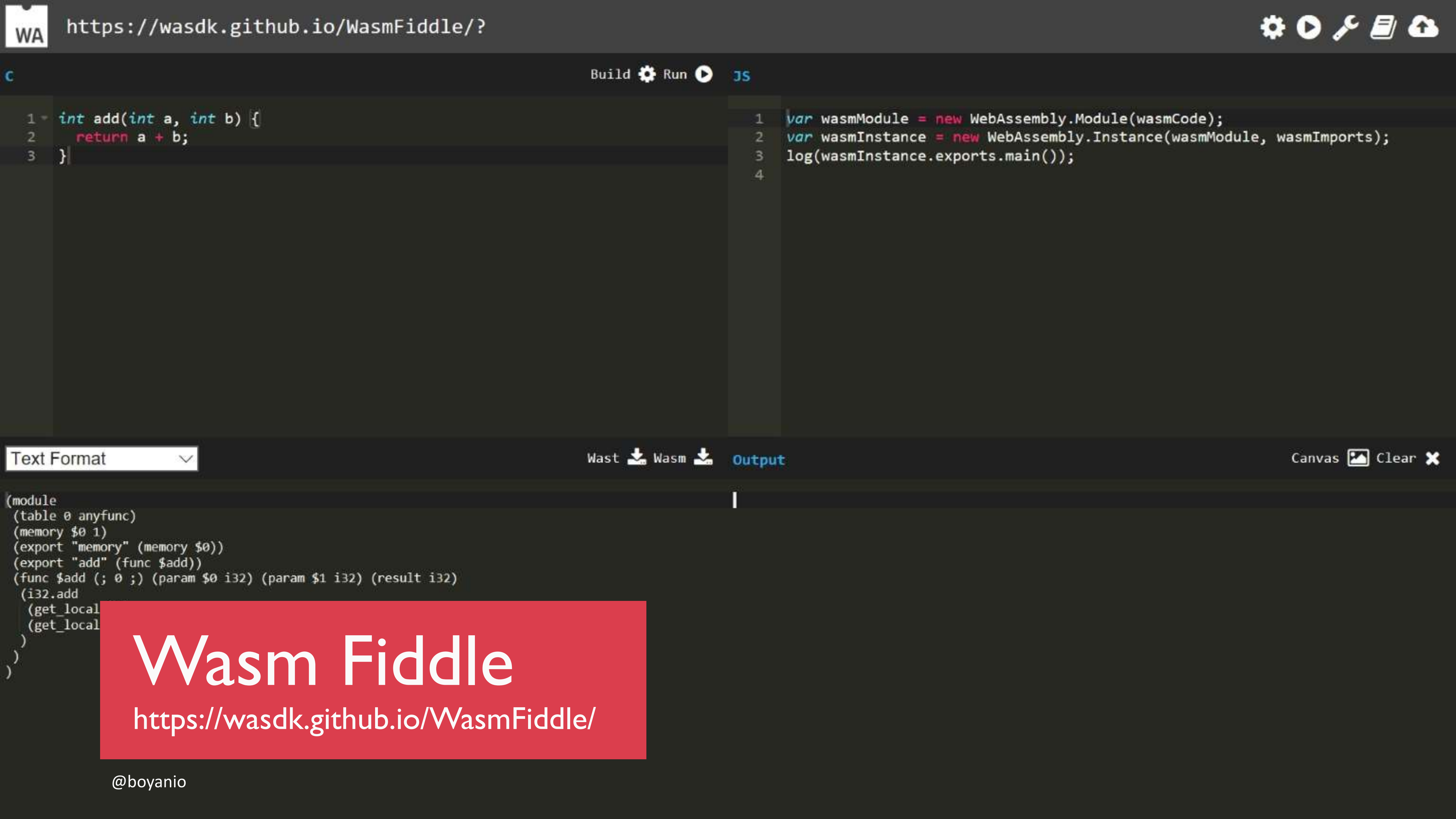
WebAssembly brings second life to your legacy code

@boyanio

https://techcrunch.com/2016/07/05/lzlabs-launches-product-to-move-mainframe-cobol-code-to-linux-cloud/

How to produce WebAssembly





```c
int add(int a, int b) {
  return a + b;
}
```

```js
var wasmModule = new WebAssembly.Module(wasmCode);
var wasmInstance = new WebAssembly.Instance(wasmModule, wasmImports);
log(wasmInstance.exports.main());
```

```
(module
 (table 0 anyfunc)
 (memory $0 1)
 (export "memory" (memory $0))
 (export "add" (func $add))
 (func $add (; 0 ;) (param $0 i32) (param $1 i32) (result i32)
  (i32.add
   (get_local
   (get_local
  )
 )
)
```

Wasm Fiddle

https://wasdk.github.io/WasmFiddle/

@boyanio

emscripten

Open Source LLVM to JavaScript compiler

@boyanio

```
emcc index.c –s WASM=1 –o index.js
```

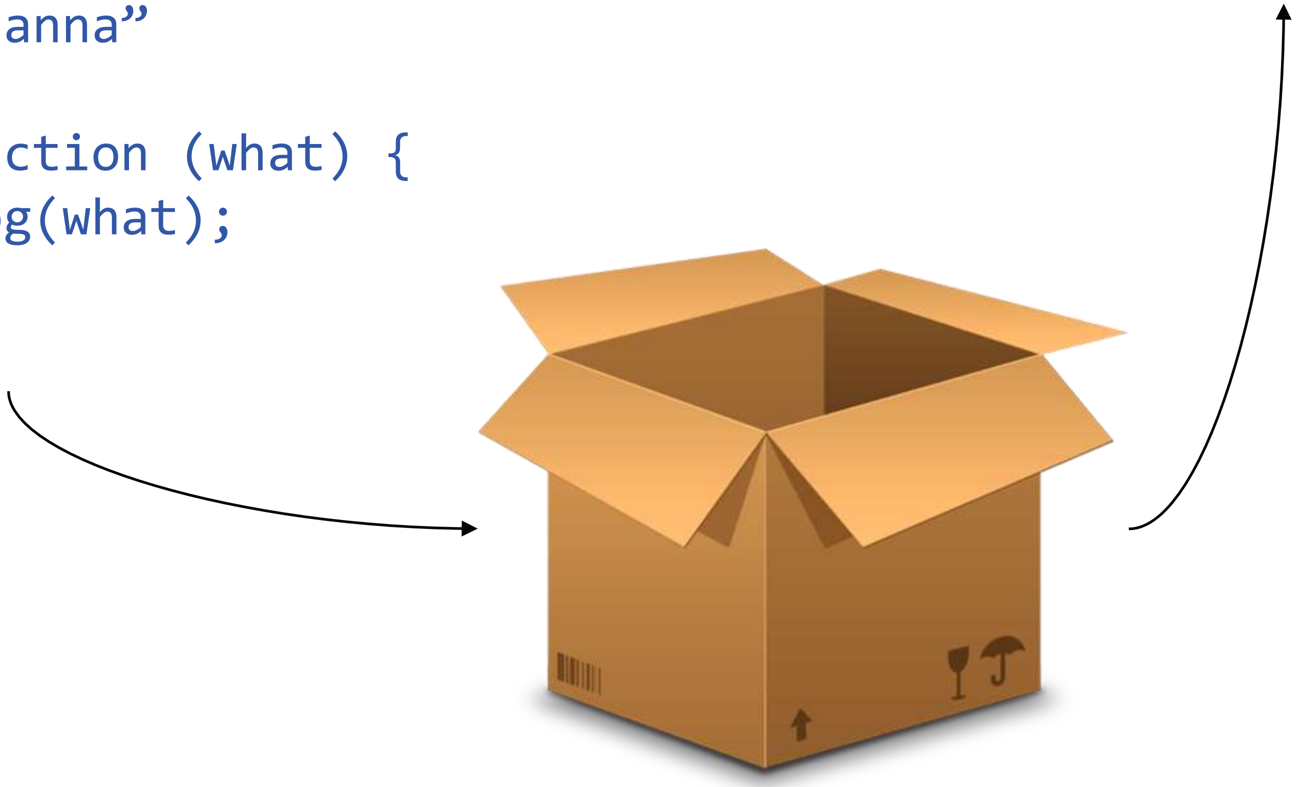
The distributable, loadable, and executable unit of code in WebAssembly is called a **module**.

https://github.com/WebAssembly/design/blob/master/Modules.md

Module imports & exports

```
const imports = {
  “name”: {
    “first”: “Anna”,
    “last”: “Nanna”
  },
  “print”: function (what) {
    console.log(what);
  }
};
```

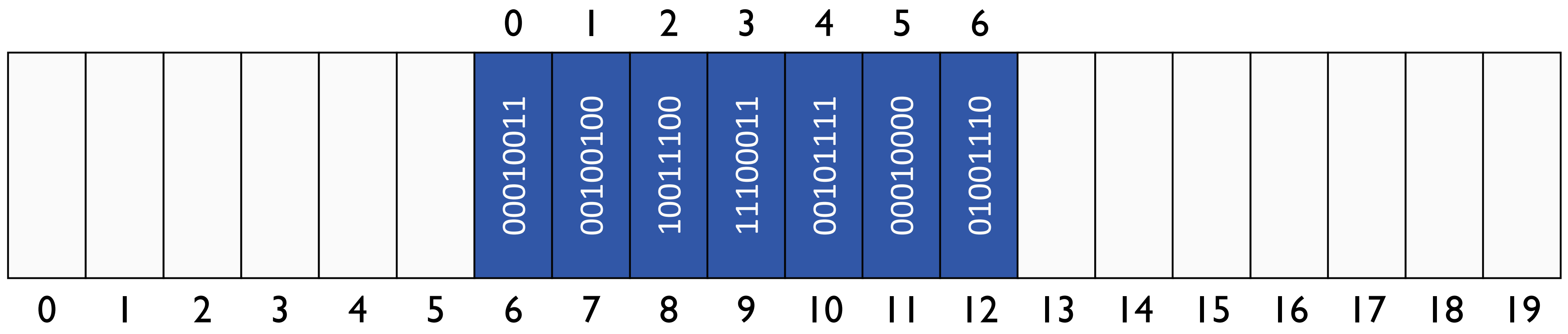
```
const exports = module.exports;
exports.printName();
exports.reverseName();
```



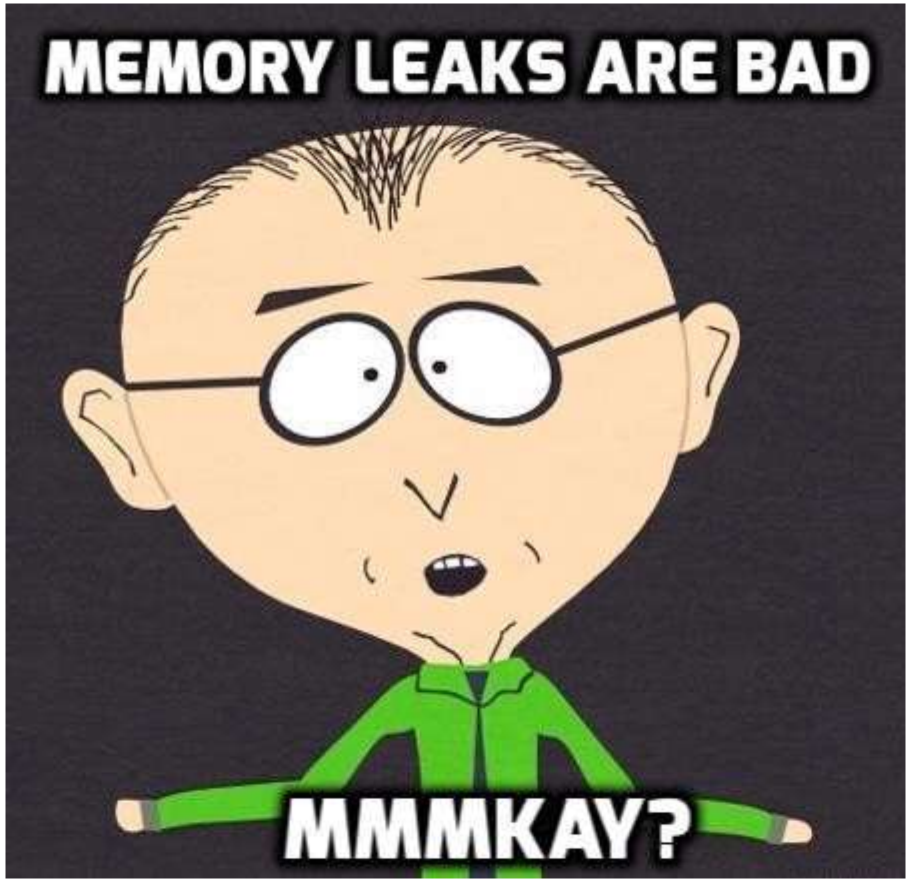
@boyanio

Linear memory

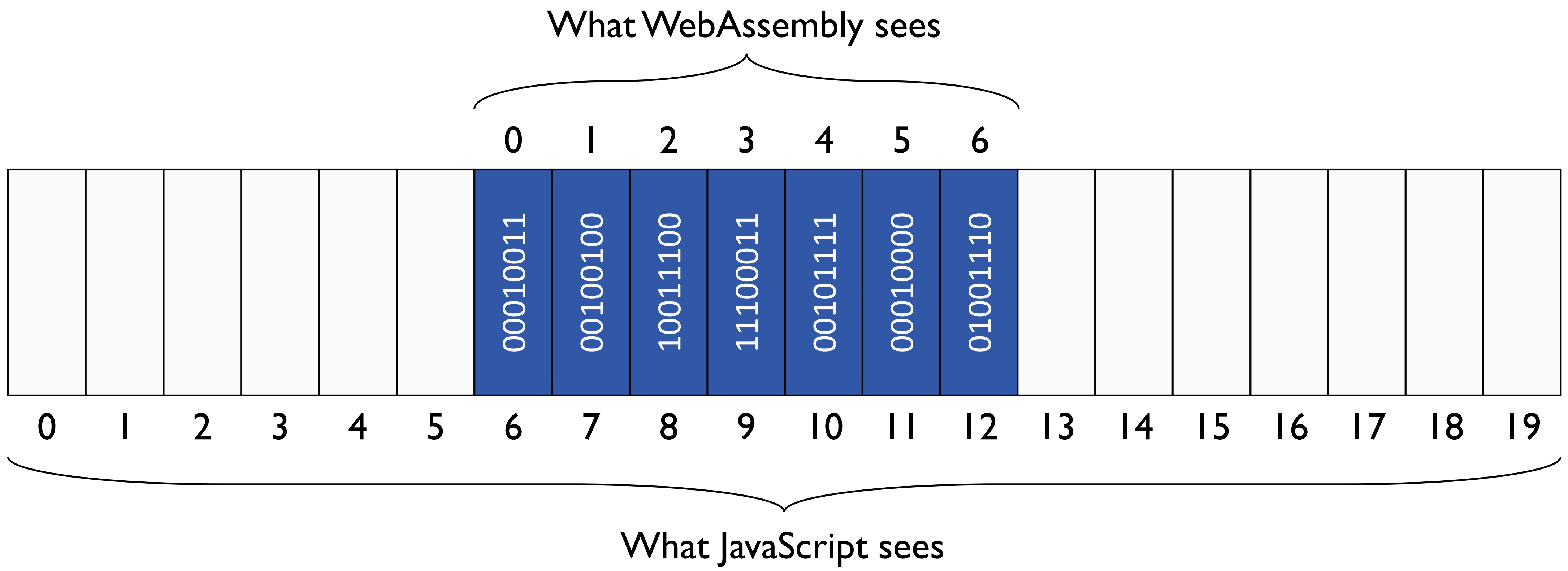
```
const imports = {
  “env”: {
    “memory”: new WebAssembly.Memory({ initial: 10, maximum: 100}),
    …
  }
};
```



@boyanio


MEMORY LEAKS ARE BAD
MMMKAY?

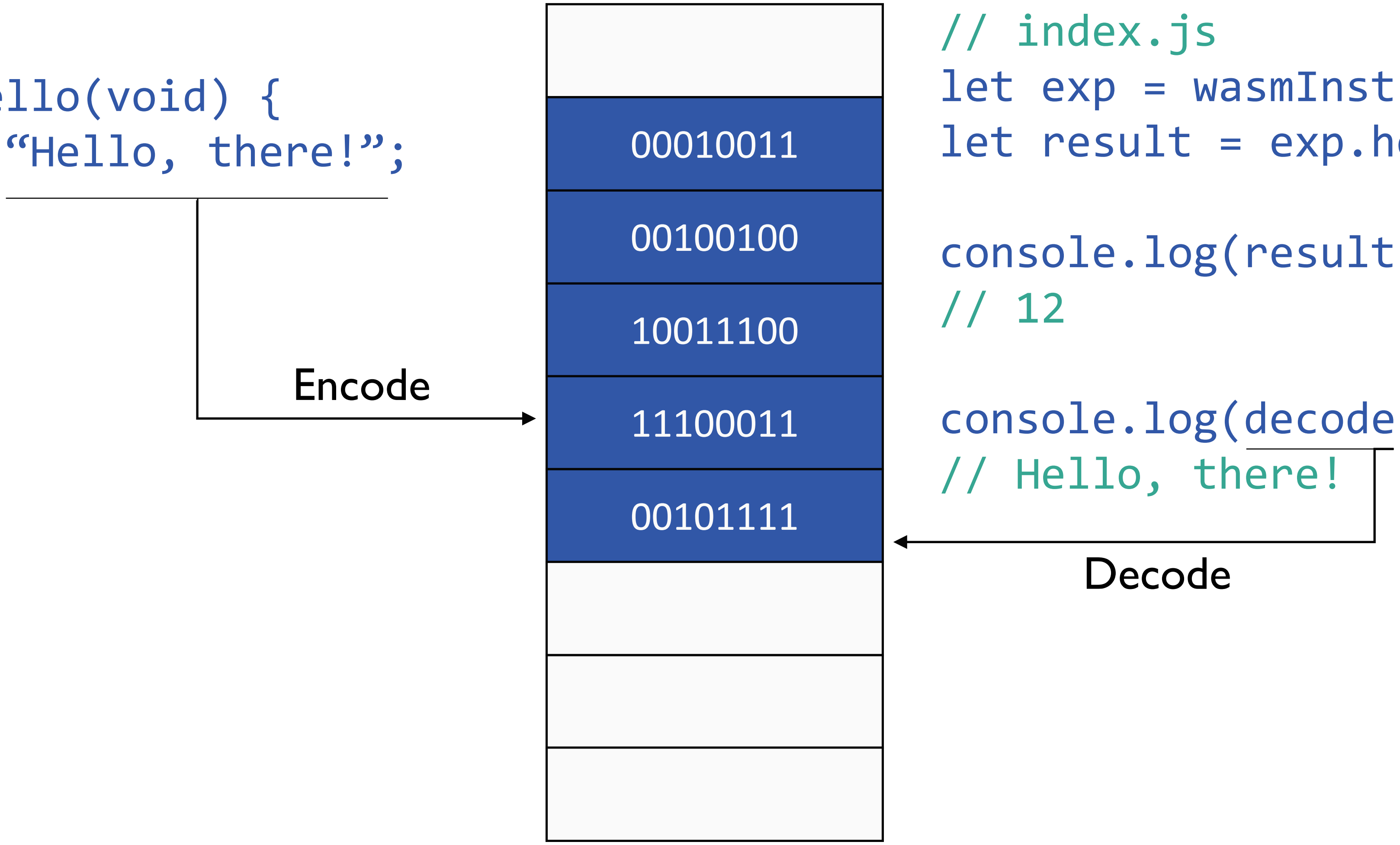
Linear memory



@boyanio

Working with strings

```
// app.c
char * hello(void) {
  return “Hello, there!”;
}
```



```
// index.js
let exp = wasmInst.exports;
let result = exp.hello();

console.log(result);
// 12

console.log(decode(result));
// Hello, there!
```

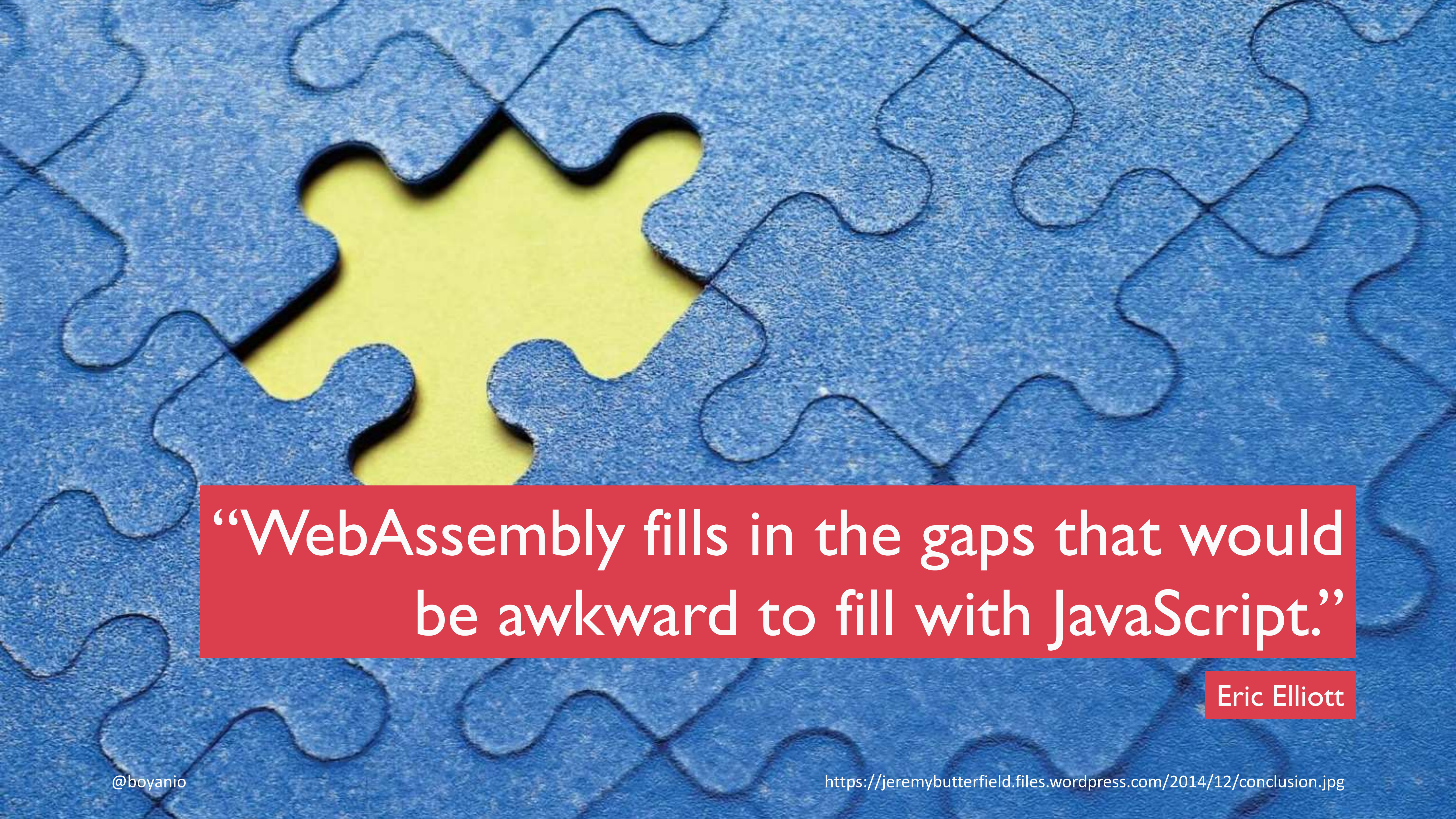
Loading WebAssembly

```
// Traditional approach

fetch('app.wasm')
  .then(result => result.arrayBuffer())
  .then(buffer => WebAssembly.instantiate(buffer, imports))
  .then(({ module, instance }) => {
    instance.exports.main();
  });

// Using the streaming API

WebAssembly.instantiateStreaming(fetch('app.wasm'), imports)
  .then(({ module, instance }) => {
    instance.exports.main();
  });
```

WASM REPLACING JAVASCRIPT?

22:57 WILL WEBASSEMBLY OVERTAKE JAVASCRIPT IN WEB APPLICATION CODING NEEDS?

https://www.washingtonexaminer.com/cnn-nyt-reporters-aggressively-miss-the-point-with-nikki-haleys-reaction-to-the-grammys-stupid-fire-and-fury-reading



“WebAssembly fills in the gaps that would be awkward to fill with JavaScript.”

Eric Elliott

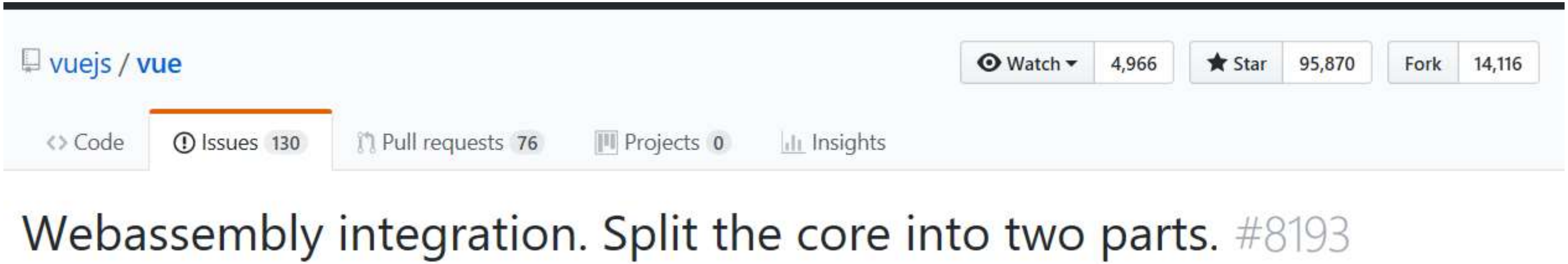
https://jeremybutterfield.files.wordpress.com/2014/12/conclusion.jpg

What about rewriting existing Web frameworks in WebAssembly?

No direct DOM access at the moment.
Do you really want to program in C/C++?
But…

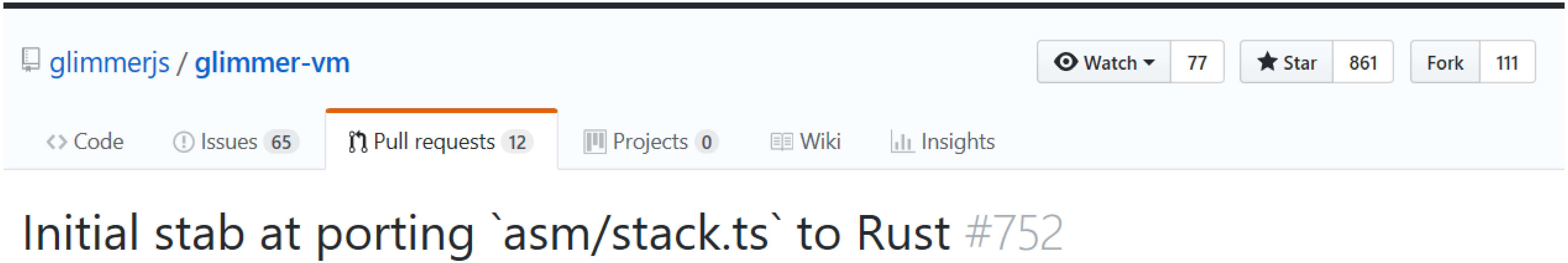
Rewriting existing Web frameworks in WebAssembly could benefit native compilation

ionic React Native NativeScript



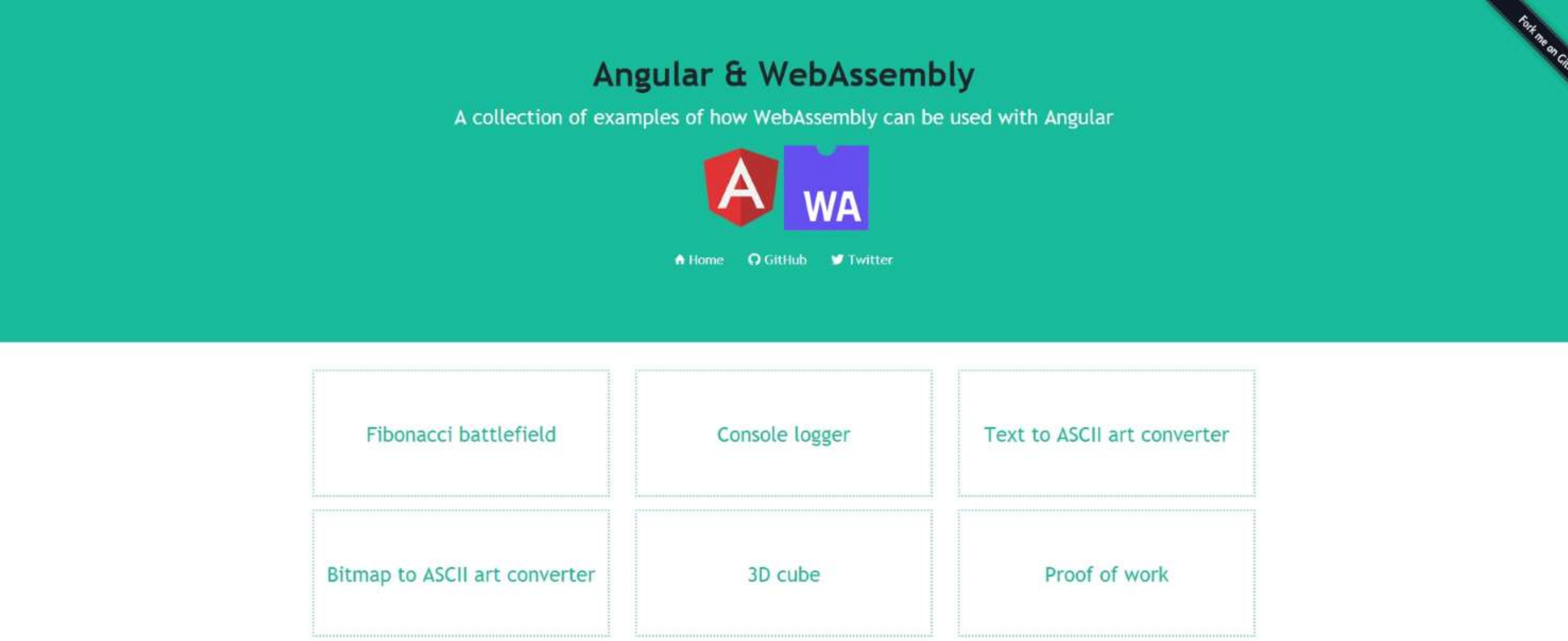
Webassembly integration. Split the core into two parts. #8193

https://github.com/vuejs/vue/issues/8193



Initial stab at porting `asm/stack.ts` to Rust #752

https://github.com/glimmerjs/glimmer-vm/pull/752


Angular & WebAssembly

A collection of examples of how WebAssembly can be used with Angular

Fibonacci battlefield

Console logger

Text to ASCII art converter

Bitmap to ASCII art converter

3D cube

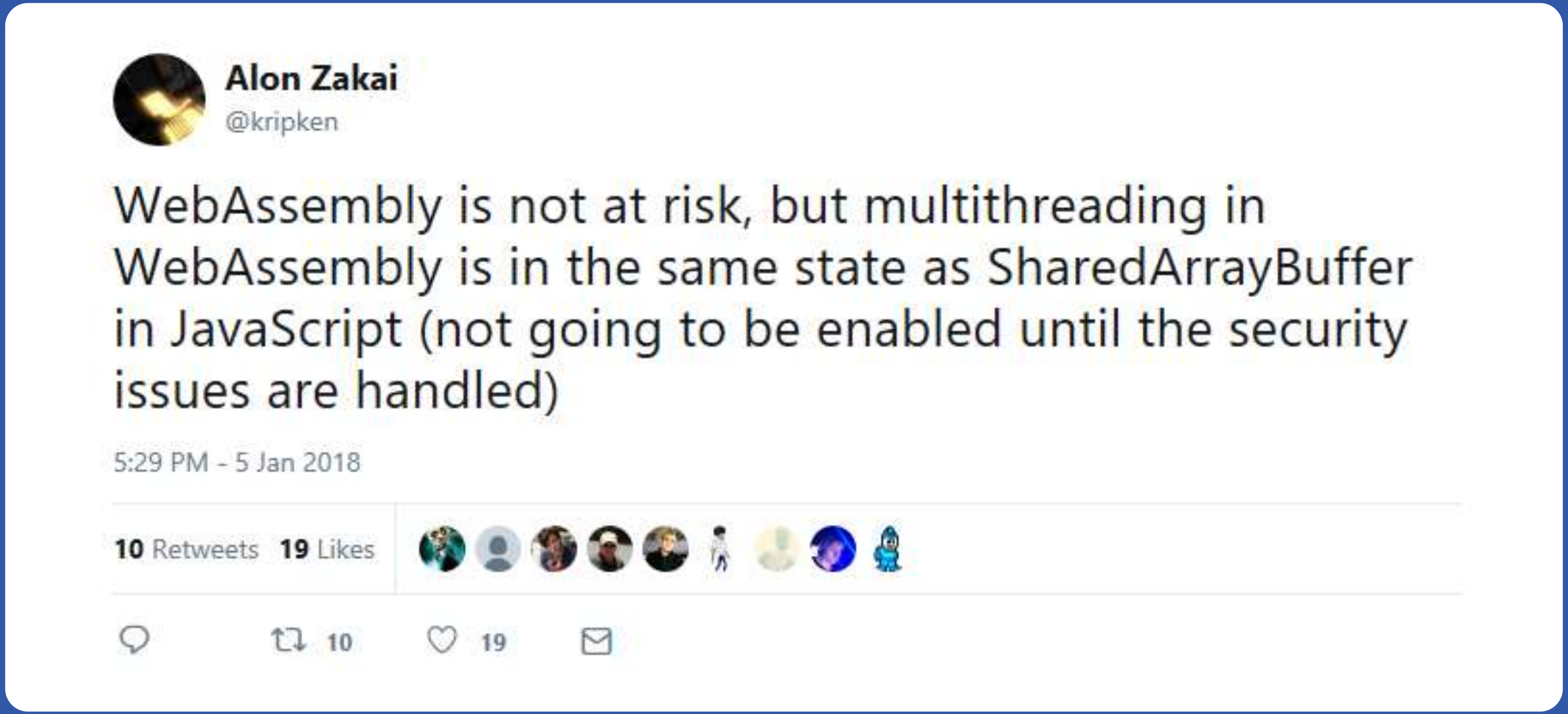
Proof of work

Angular & WebAssembly

https://boyan.io/angular-wasm/

How secure is WebAssembly?

@boyanio



Alon Zakai
@kripken

WebAssembly is not at risk, but multithreading in WebAssembly is in the same state as SharedArrayBuffer in JavaScript (not going to be enabled until the security issues are handled)

5:29 PM - 5 Jan 2018

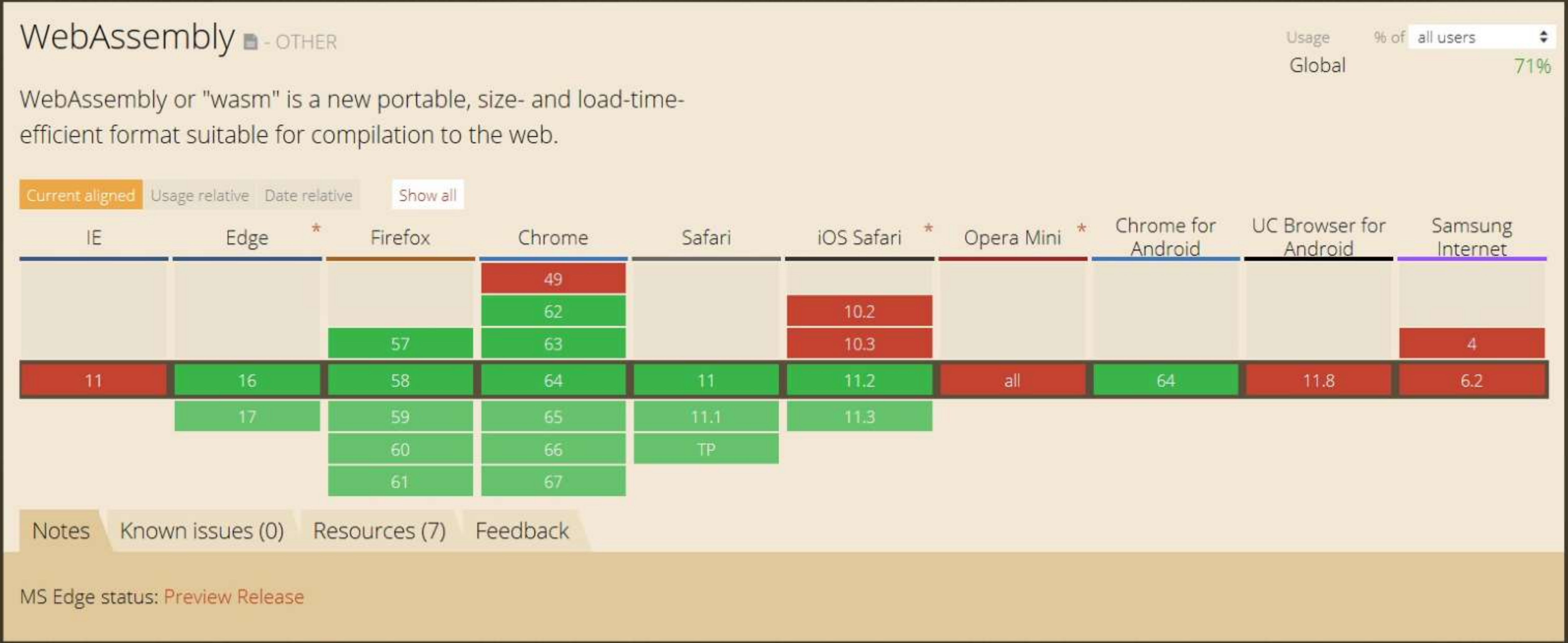
10 Retweets **19** Likes

10 19

@boyanio

WebAssembly runs in a memory-safe sandboxed environment

Can I use WebAssembly ?



WebAssembly - OTHER

Usage % of all users
Global 71%

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

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			🟥 49						
			🟩 62		🟥 10.2				
		🟩 57	🟩 63		🟥 10.3				🟥 4
🟥 11	🟩 16	🟩 58	🟩 64	🟩 11	🟩 11.2	🟥 all	🟩 64	🟥 11.8	🟥 6.2
	🟩 17	🟩 59	🟩 65	🟩 11.1	🟩 11.3				
		🟩 60	🟩 66	🟩 TP					
		🟩 61	🟩 67						

Notes | Known issues (0) | Resources (7) | Feedback

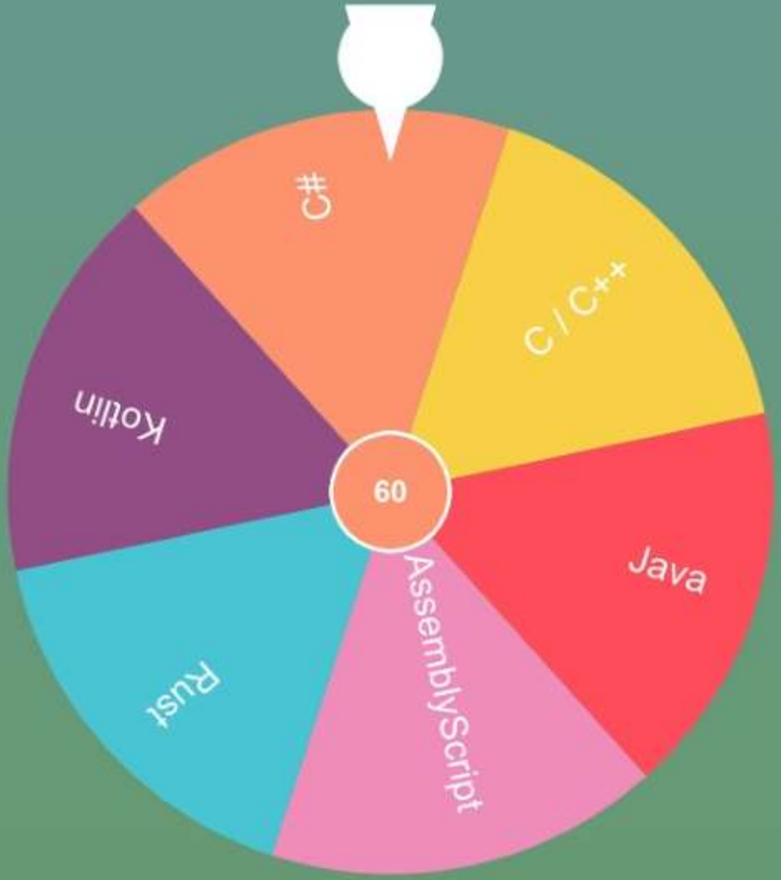
MS Edge status: Preview Release

@boyanio

http://caniuse.com/#search=WebAssembly

What languages can be compiled to WebAssembly?

https://boyan.io/wasm-wheel/



What’s next?

Direct access to the DOM and WebAPIs
Integration with browser’s GC
Multi-threading (maybe)

The future of Web belongs to those, who compile

Boyan Mihaylov / @boyanio / boyan.io