

WebAssembly brings the Web to a new era

Boyan Mihaylov
@boyanio
boyan.io

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

```
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!
```

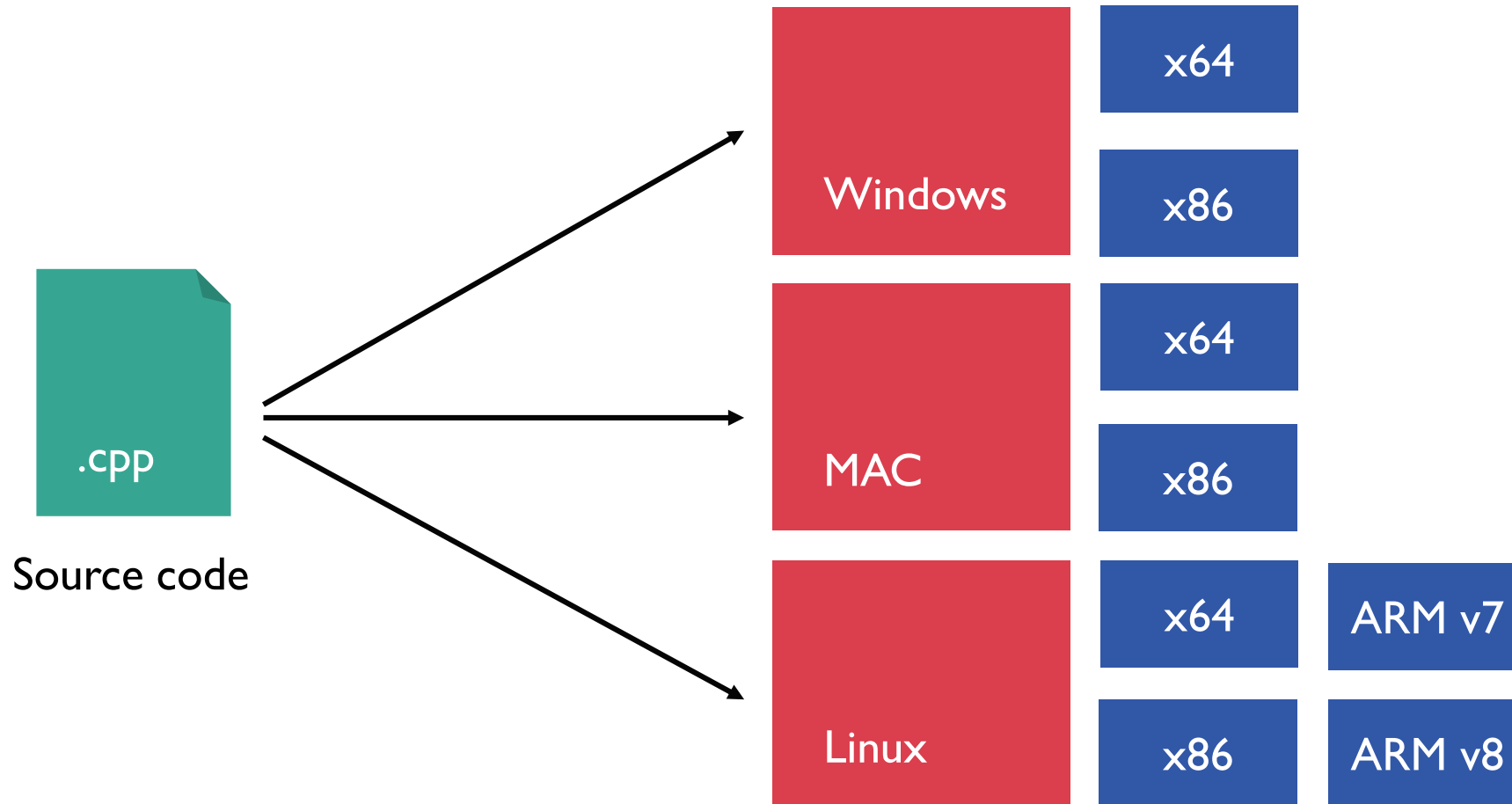
```
D:\wasm>emcc index.c -s WASM=1 -o a.js
```

```
D:\wasm>node a.js
```

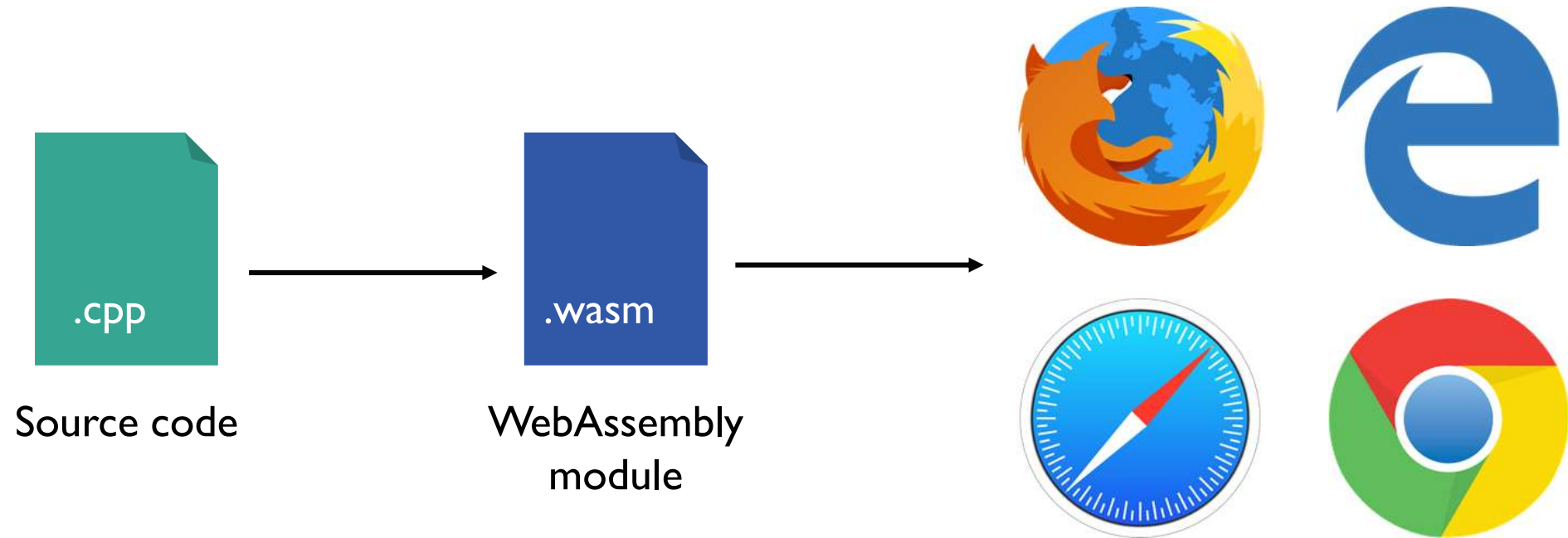
```
Hello, cool people!
```

```
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"

```
> '7' - 3
```

```
< 4
```

weak typing, implicit conversion

```
> '7' + 3
```

```
< "73"
```

...not really consistent

```
> '7' - '3'
```

```
< 4
```

string - string = number ?

```
> 7 + '3'
```

```
< "73"
```

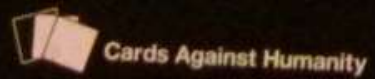
“+” is for concatenation

```
> 7 + + '3'
```

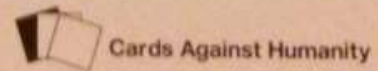
```
< 10
```

“+ +” is for addition ?

I drink to forget



JavaScript



WebAssembly is a typed language

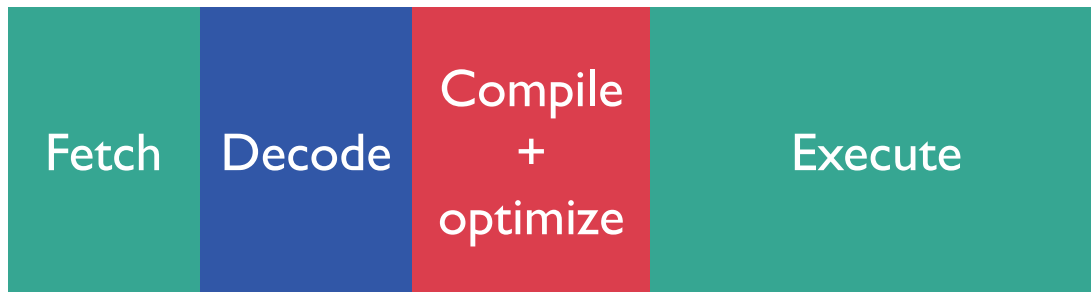
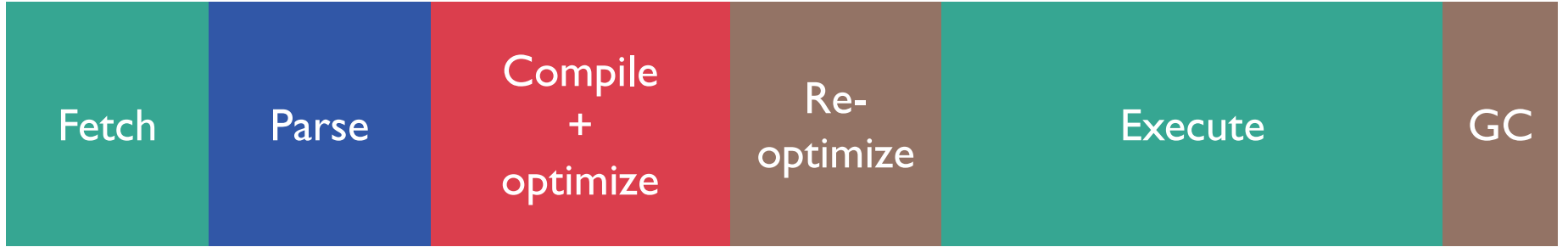
It supports 32 and 64-bit integers (i32, i64) and floating points (f32, f64)

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)
    )
  )
)
```



**WebAssembly provides consistent,
predictable performance**

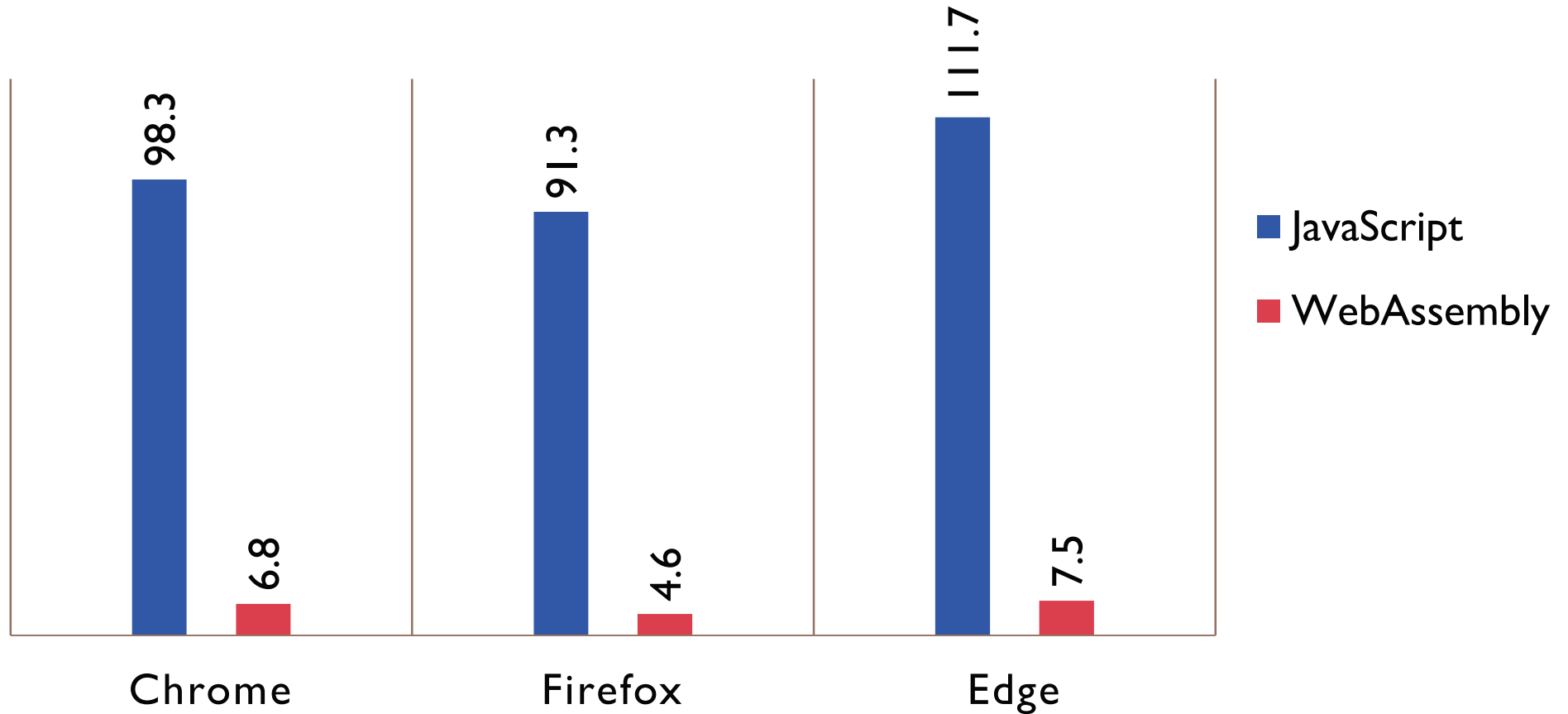


3D animation performance

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

Performance comparison

Average animation time (ms)





The Adobe Flash plugin has crashed.
[Send crash report](#)

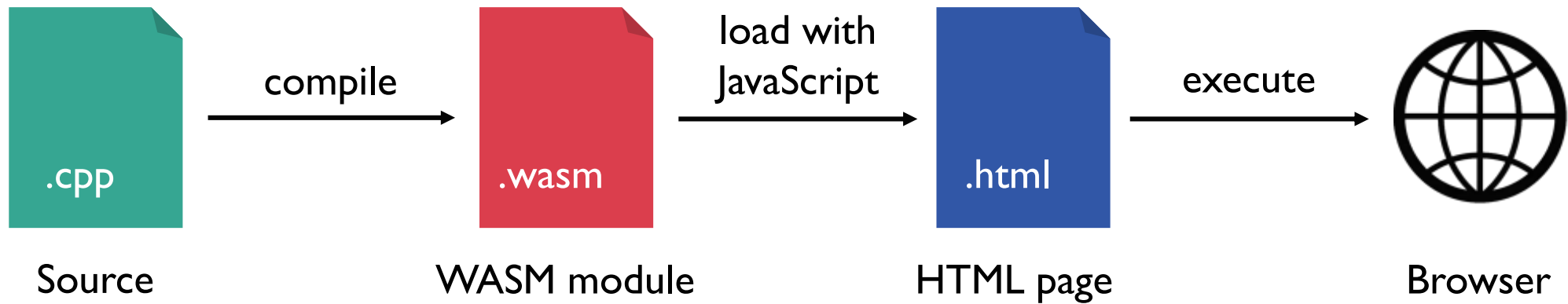
Reusing code on the Web

**WebAssembly enables code reusability
between native and Web**



WebAssembly brings second life to your legacy code

How to produce WebAssembly



C

Build ⚙️ Run ▶️ JS

```
1 int add(int a, int b) {  
2   return a + b;  
3 }
```

```
1 var wasmModule = new WebAssembly.Module(wasmCode);  
2 var wasmInstance = new WebAssembly.Instance(wasmModule, wasmImports);  
3 log(wasmInstance.exports.main());  
4
```

Text Format ▼

Wast 📄 Wasm 📄 Output

Canvas 🖼️ Clear ✕

```
(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/>



Open Source LLVM to JavaScript compiler

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

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

Module imports & exports

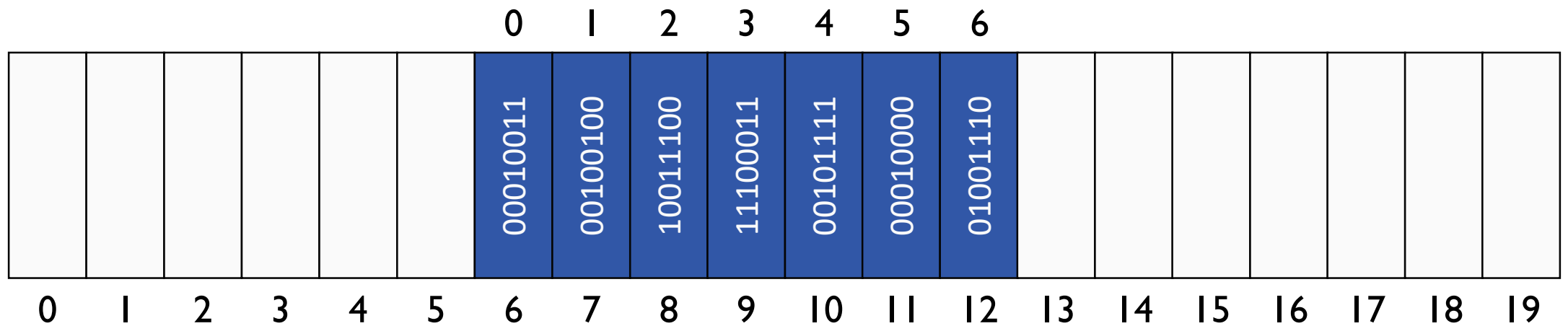
```
const imports = {  
  "name": {  
    "first": "Anna",  
    "last": "Nanna"  
  },  
  "print": function (what) {  
    console.log(what);  
  }  
};
```

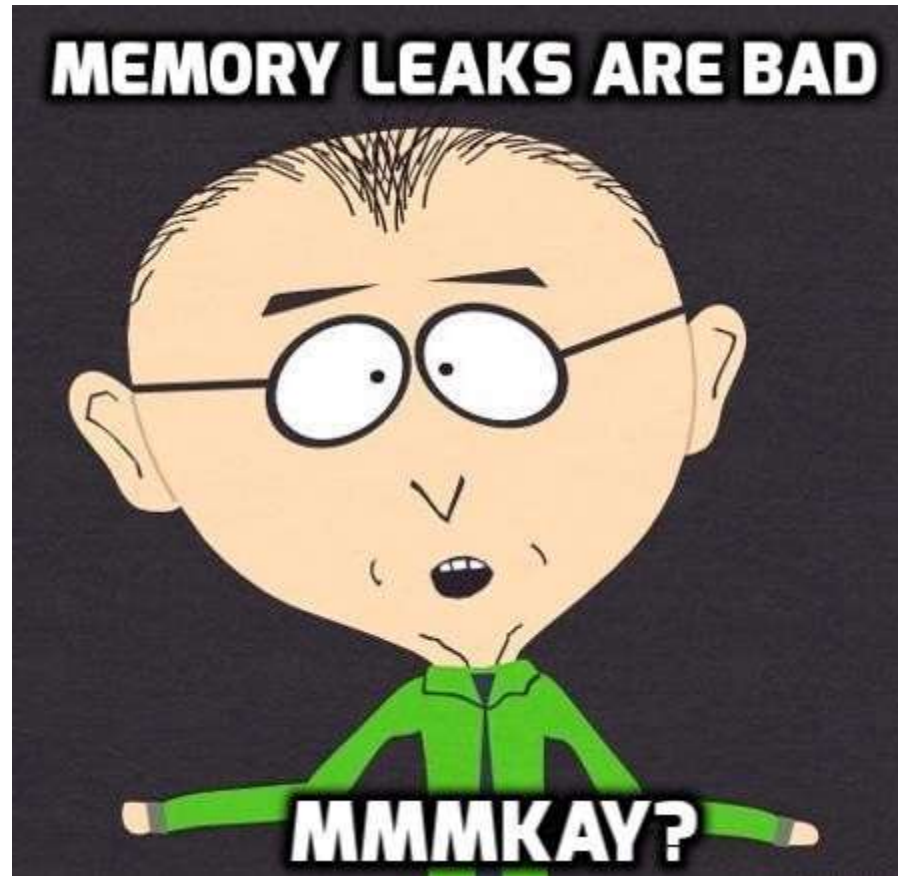
```
const exports = module.exports;  
exports.printName();  
exports.reverseName();
```



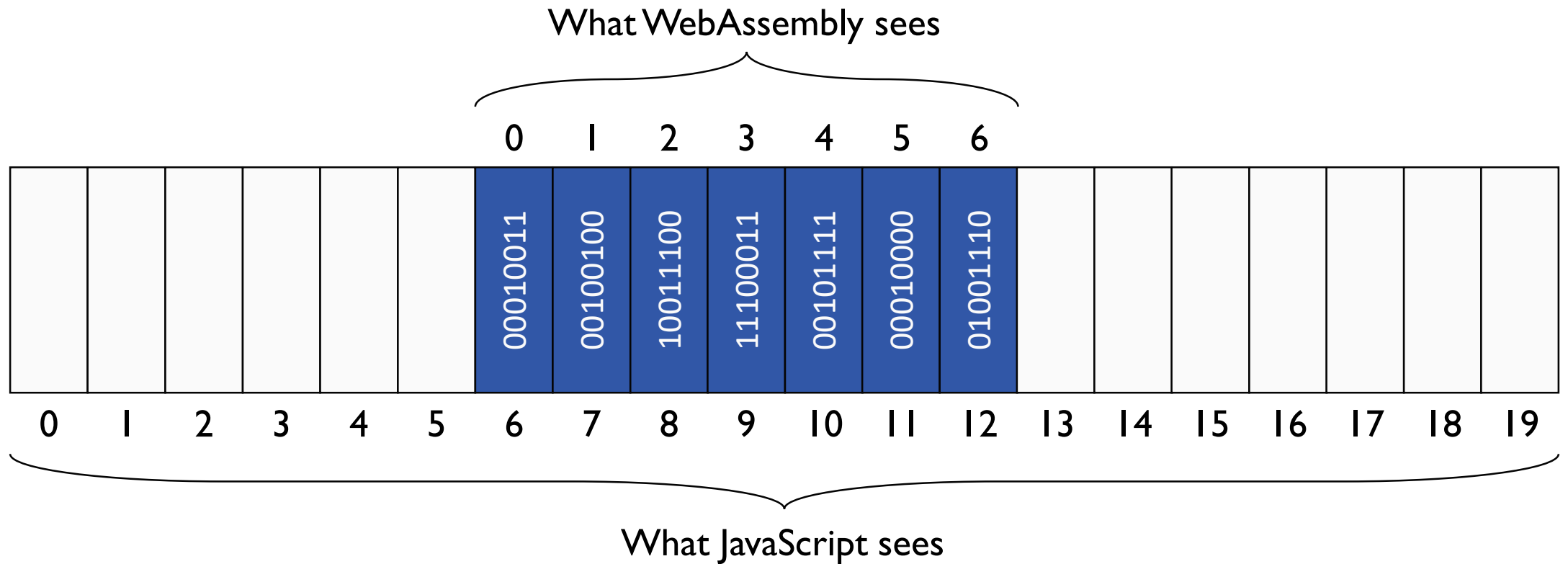
Linear memory

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





Linear memory



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();
  });
```

LIVE

BREAKING NEWS

WASM REPLACING JAVASCRIPT?

22:57

WILL WEBASSEMBLY OVERTAKE JAVASCRIPT IN WEB APPLICATION CODING NEEDS?

@boyanio

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

A close-up photograph of a blue puzzle. One piece is missing, revealing a bright yellow surface underneath. The puzzle pieces are interlocking and have a slightly textured appearance.

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

Eric Elliott

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



vuejs / vue

Watch ▾

4,966

★ Star

95,870

Fork

14,116

<> Code

! Issues 130

🔗 Pull requests 76

📁 Projects 0

📊 Insights

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

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

glimmerjs / glimmer-vm

Watch ▾

77

★ Star

861

Fork

111

<> Code

! Issues 65

🔗 Pull requests 12

📁 Projects 0

📖 Wiki

📊 Insights

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

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

How secure is WebAssembly?





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



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

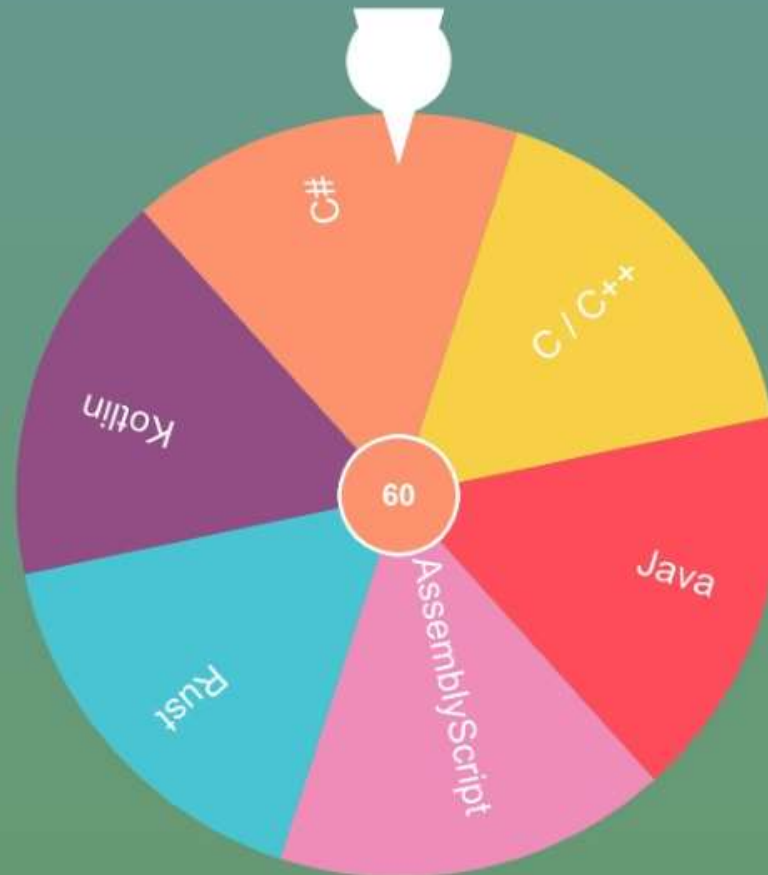
IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			62		10.2				
		57	63		10.3				4
11	16	58	64	11	11.2	all	64	11.8	6.2
	17	59	65	11.1	11.3				
		60	66	TP					
		61	67						

Notes Known issues (0) Resources (7) Feedback

MS Edge status: **Preview Release**

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