A journey of multithreading using WebAssembly

Boyan Mihaylov
@boyanio
boyan.io

“Many hands make light work”

@boyanio

https://idioms.thefreedictionary.com/many+hands+make+light+work

Partial timeline of multithreaded systems

1950s	1960s	2000s
NBS SEAC	CDC 6600	CDC 6600
DYSEAC	IBM ACS-360	IBM ACS-360
Lincoln Labs TX-2		

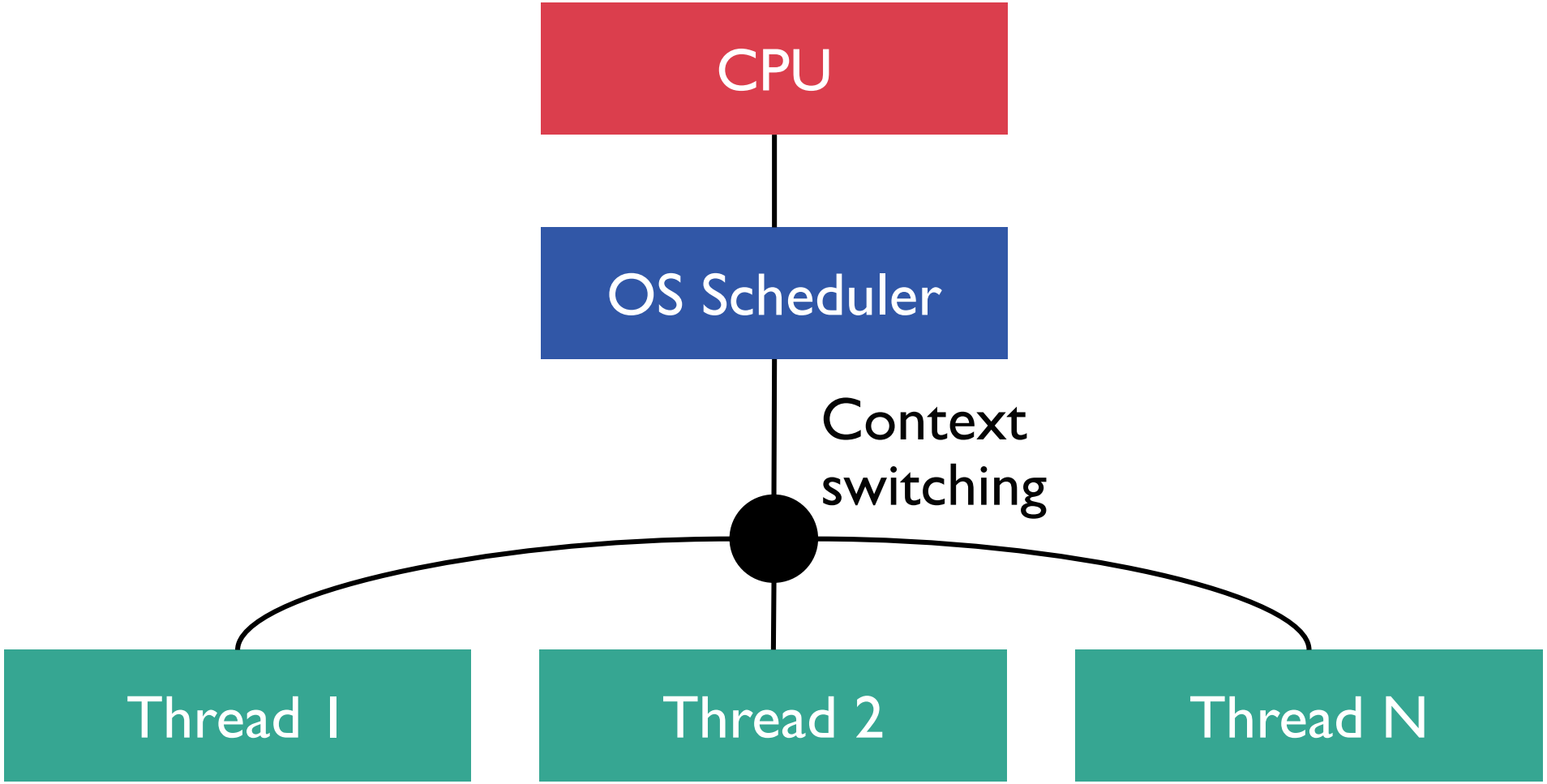
https://people.cs.clemson.edu/~mark/multithreading.html

“[…] multithreading is the ability of a central processing unit (CPU) […] to provide multiple threads of execution concurrently […]”

https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)

“[…] a thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler […]”

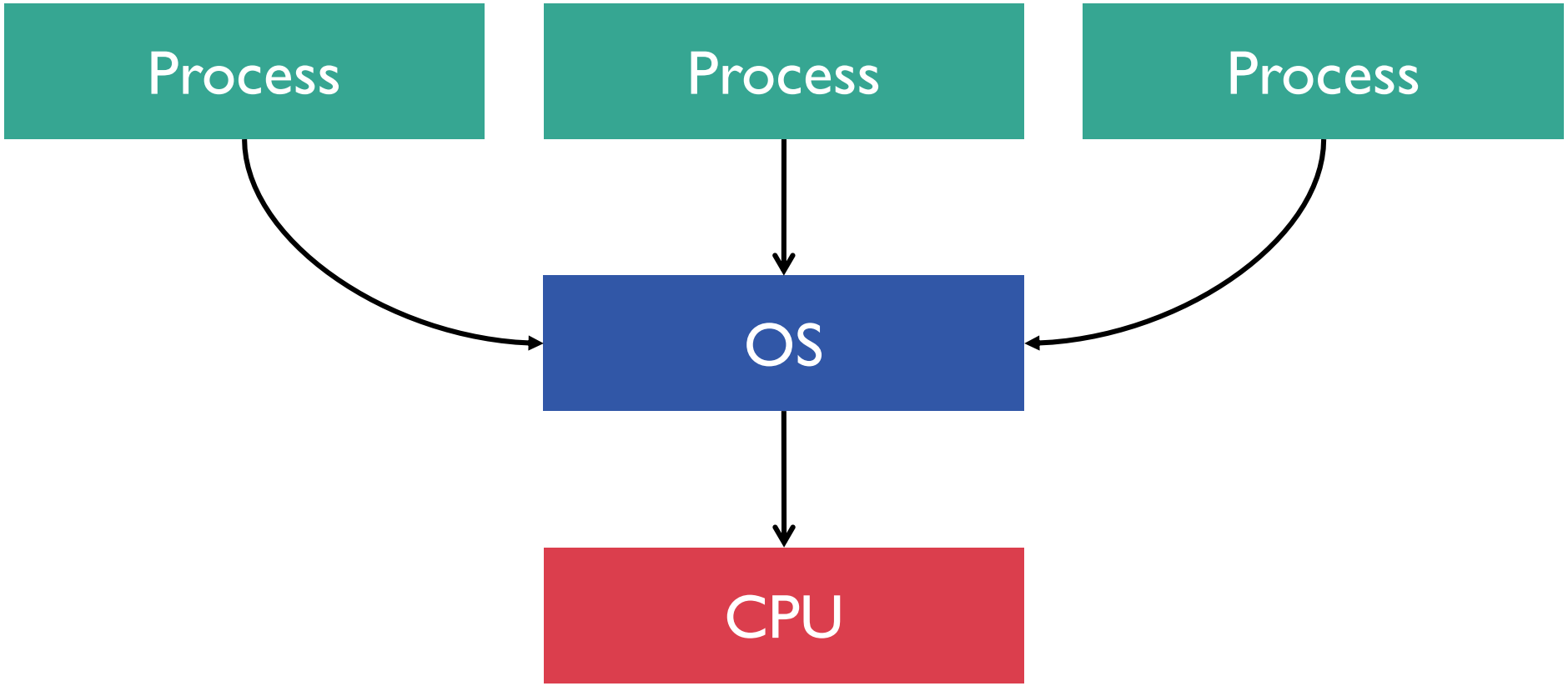
https://en.wikipedia.org/wiki/Thread_(computing)


CPU
OS Scheduler
Context switching
Thread 1
Thread 2
Thread N

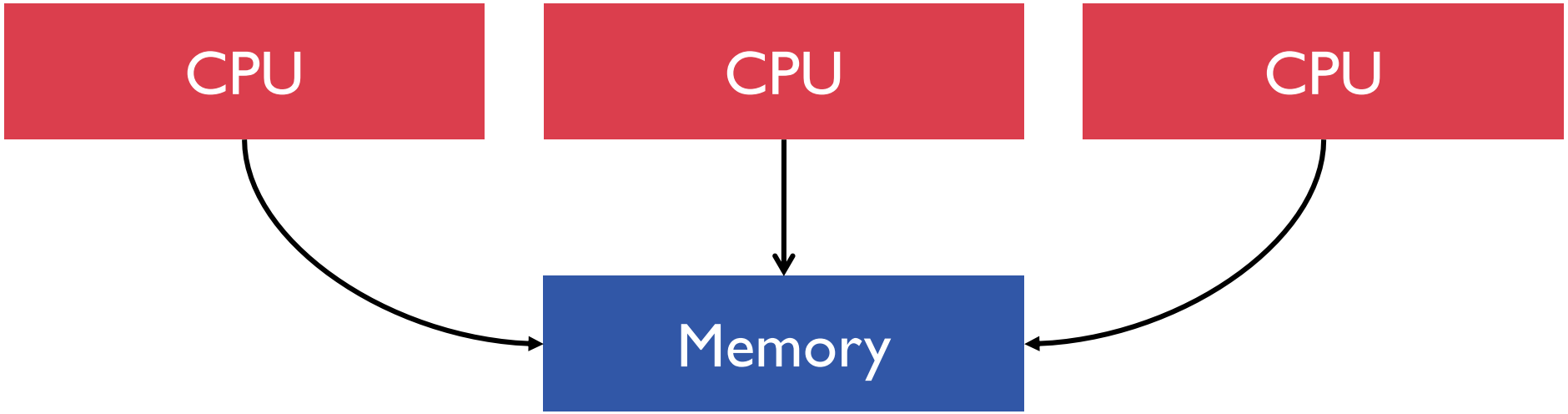
@boyanio

Multithreading
vs.
Multiprocessing
vs.
Multitasking

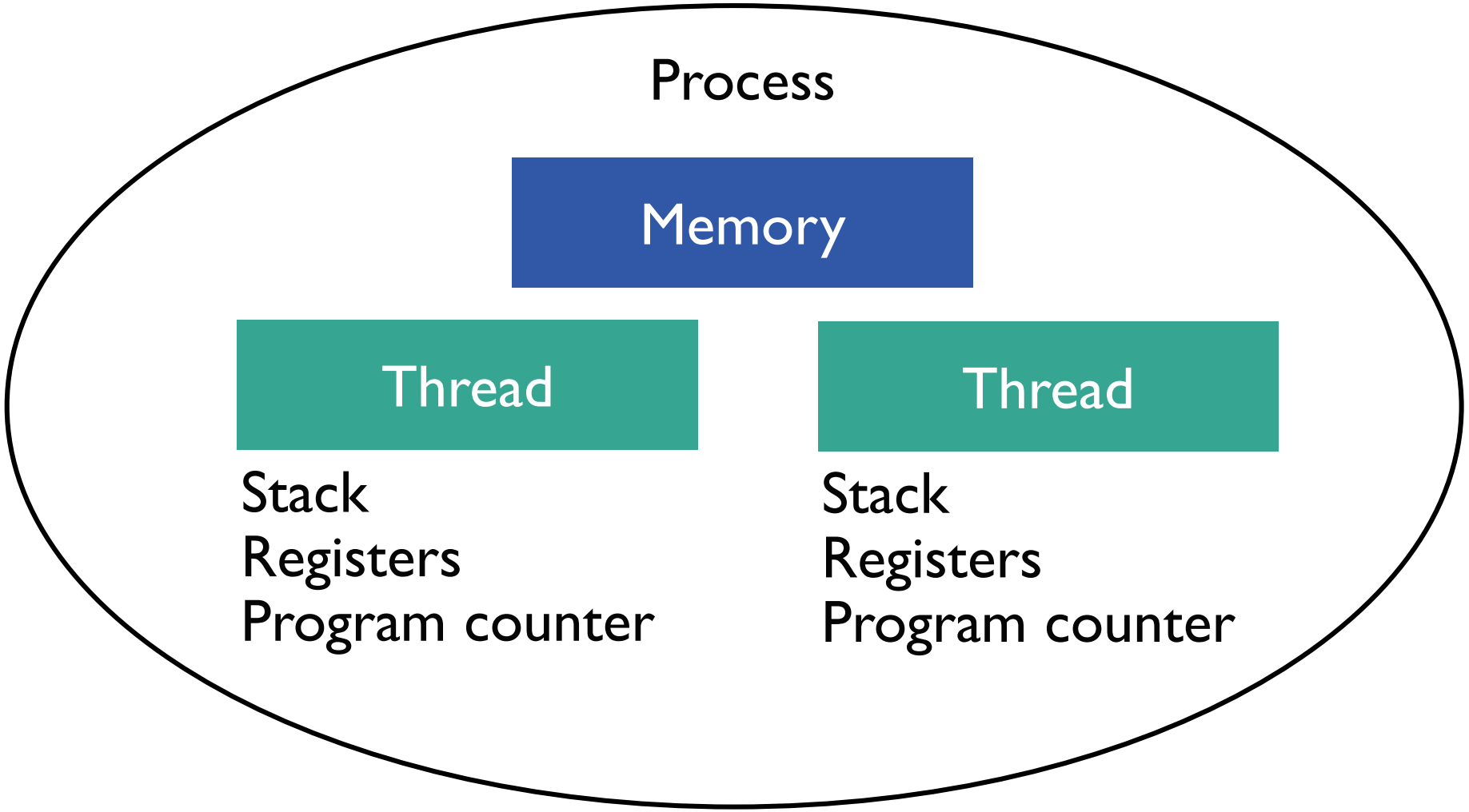
Multitasking



Multiprocessing



Multithreading



Use multithreading

Your code has a lot of I/O or Network usage

You have a GUI

Use multiprocessing

Your code is CPU bound

Ned Batchelder
@nedbat

Some people, when confronted with a problem, think, "I know, I'll use threads," and then two they hav erpoblesms.

6:47 PM · Apr 23, 2012 · Twitter Web Client

1.6K Retweets **380** Likes

@boyanio

https://twitter.com/nedbat/status/194452404794691584

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

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

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

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

C:\wasm>emcc -o a.js index.c

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

Can we use multithreading with WebAssembly?

JavaScript is single-threaded

Calls to WebAssembly are blocking

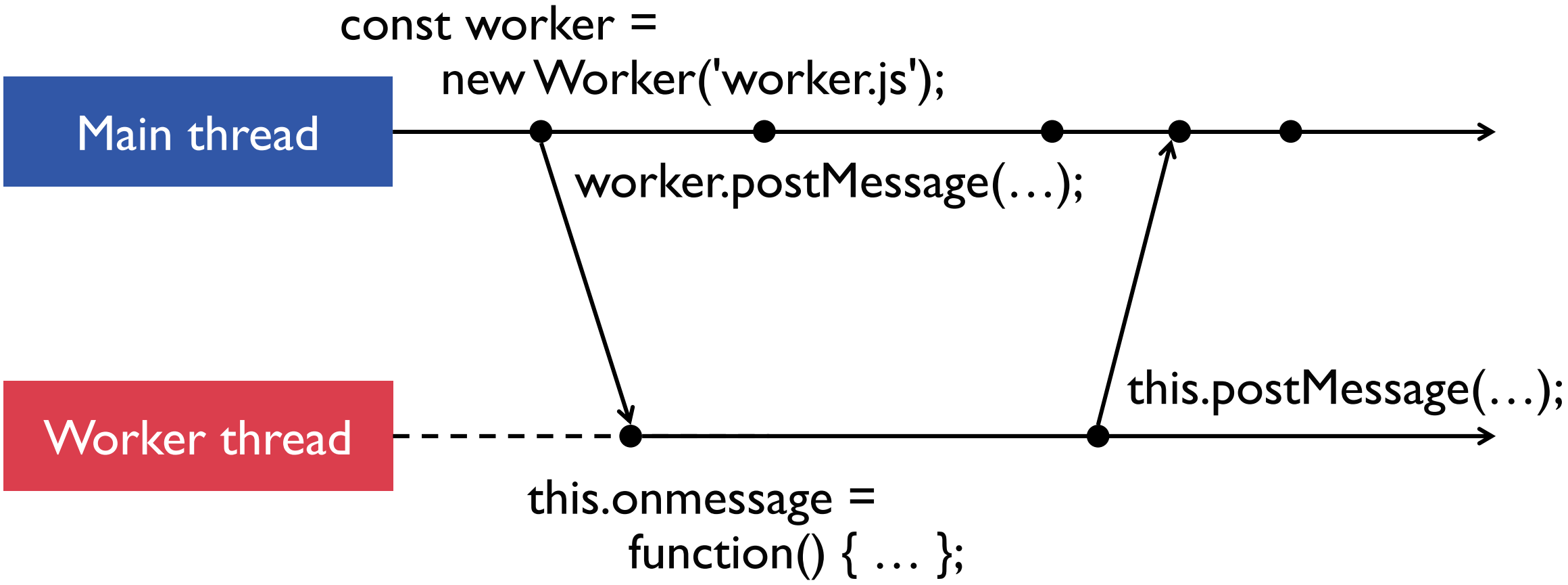
```
const wasm = await fetch('app.wasm');
const buffer = await wasm.arrayBuffer();
const { instance } =
     await WebAssembly.instantiate(buffer);

// This is a blocking call
instance.exports.calculate();
```

Web Workers are a simple means for web content to run scripts in background threads.

https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers

Web Workers


const worker =
new Worker('worker.js');
Main thread
worker.postMessage(…);
this.postMessage(…);
Worker thread
this.onmessage =
function() { … };

When a message is passed between the main thread and worker, it is cloned, not shared.

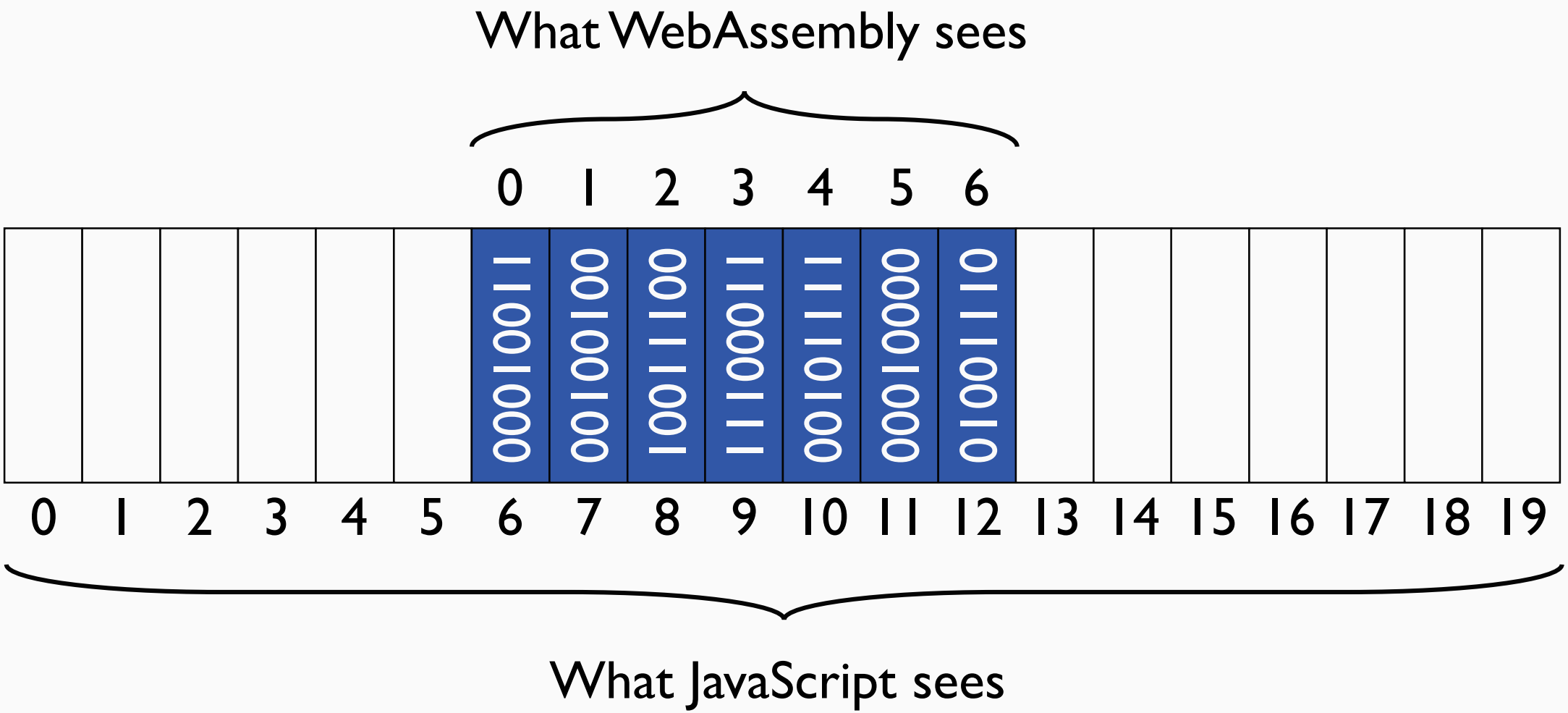
Browsers implement WebAssembly 1.0 (MVP).

Post-MVP features are on the roadmap.

WebAssembly threads proposal:

Shared linear memory
Atomic operations
Wait / Notify operators

WebAssembly linear memory



@boyanio

Creating linear memory in JavaScript

```
const memory = new WebAssembly.Memory({
    initial: 1,
    maximum: 1
});
const imports = {
    env: { memory }
};
const { instance } =
    await WebAssembly.instantiate(buffer, imports);
```

Using linear memory in WebAssembly

```
(module
    ;; Import 1 page (64Kib) of memory
    (import "env" "memory" (memory 1 1))
    …
)
```

Creating shared linear memory in JavaScript

```
const memory = new WebAssembly.Memory({
    initial: 1,
    maximum: 1,
    shared: true
});
const imports = {
    env: { memory }
};
const { instance } =
    await WebAssembly.instantiate(buffer, imports);
```

Using shared linear memory in WebAssembly

```
(module
    ;; Import 1 page (64Kib) of shared memory
    (import "env" "memory" (memory 1 1 shared))
    …
)
```

WebAssembly shared memory uses **SharedArrayBuffer** under the hood

Web Workers with SharedArrayBuffer

Main thread

```
const buffer = new SharedArrayBuffer(8);
const view = new Uint8Array(buffer);
view[0] = 1;
const worker = new Worker('worker.js');
worker.postMessage({ buffer });
```

Worker thread

```
this.onmessage = ({ data }) => {
    const buffer = data.buffer;
    const view = new Uint8Array(buffer);
    // Prints 1
    console.log(view[0]);
};
```

@boyanio

Security concerns of shared memory

Alon Zakai
@kripken

Replying to @WasmWeekly

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)

6:29 PM · Jan 5, 2018 · Twitter Web Client

https://twitter.com/kripken/status/949317013184290817

“SharedArrayBuffer is now re-enabled in Chrome versions where Site Isolation is on by default.”

https://www.chromium.org/Home/chromium-security/ssca

Shared Array Buffer - OTHER

Usage % of all users ?

Global 30.75%

Type of ArrayBuffer that can be shared across Workers.

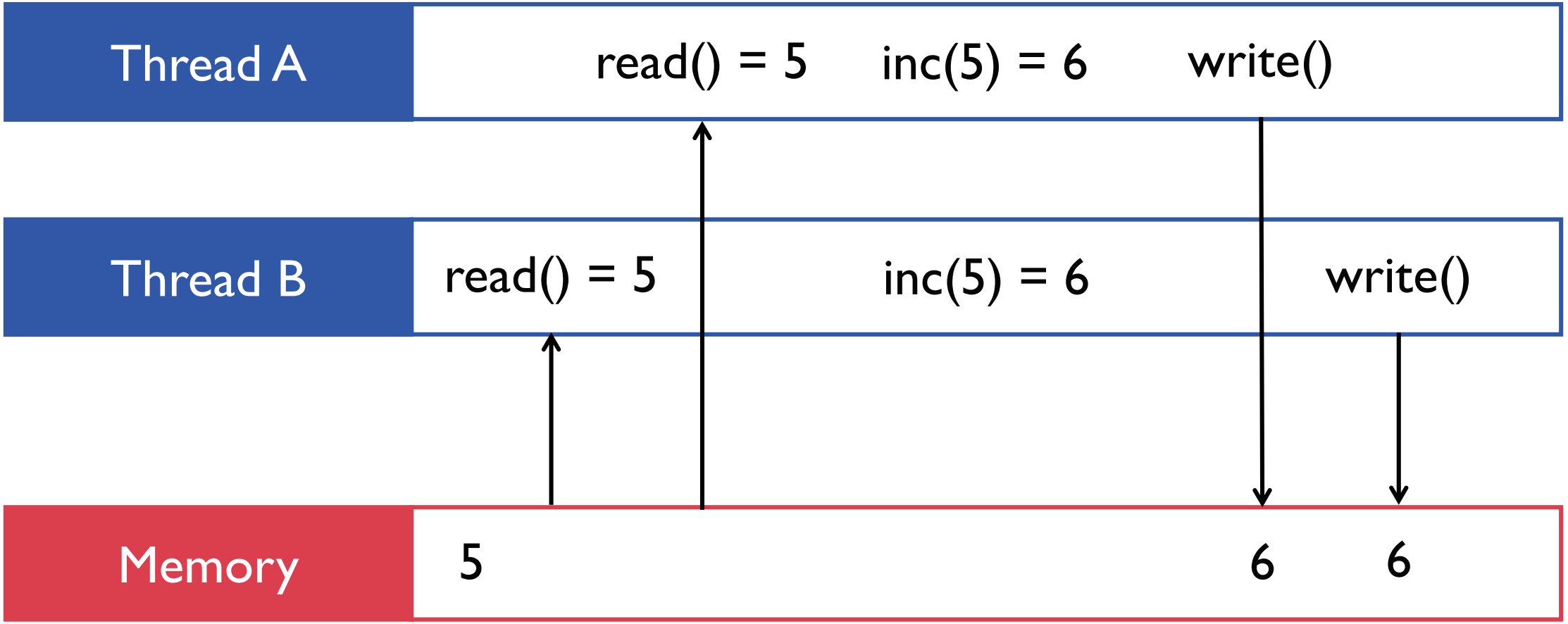
Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	C
			4-59											
	12-15	2-56	[1] 60-67	3.1-10	10-46	3.2-10.2								
6-10	[1] 16-17	[1] 57-68	68-76	[1] 10.1-12.1	[1] 47-60	[1] 10.3-12.3		2.1-4.4.4	12-12.1				4-8.2	
11	[1] 18	[1] 69	77	[1] 13	[1] 62	[1] 13	all	76	46	[1] 76	[1] 68	12.12	9.2	
	76	[1] 70-71	78-80	[1] 13.1-TP		[1] 13.1								

Notes | Known issues (0) | Resources (4) | Feedback

[1] Has support, but was disabled across browsers in January 2018 due to Spectre & Meltdown vulnerabilities.

Why we need atomic operations



@boyanio

Atomic operations

- Load / Store
- Read-Modify-Write
- Compare exchange

Wait / Notify operators

```
;; Wait for the other agent to finish with mutex
(i32.atomic.wait
  (local.get $addr)    ;; mutex address
  (i32.const 1)        ;; expected value: 1 -> locked
  (i64.const -1))      ;; timeout, infinite
```

```
;; Notify agents that are waiting on this lock
(atomic.notify
  (local.get $addr)    ;; mutex address
  (i32.const 1))       ;; notify up to 1 waiter
```

Wasm Workers

WebAssembly shared linear memory with atomic operations in Web Workers

Fork me on GitHub

Completed in 2498ms

Worker 0 colored 19 cells
Worker 1 colored 19 cells
Worker 2 colored 18 cells
Worker 3 colored 7 cells
Worker 4 colored 18 cells

Total 81 cells colored

Matrix width: Matrix height: Workers:

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Wasm Workers

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

@boyanio

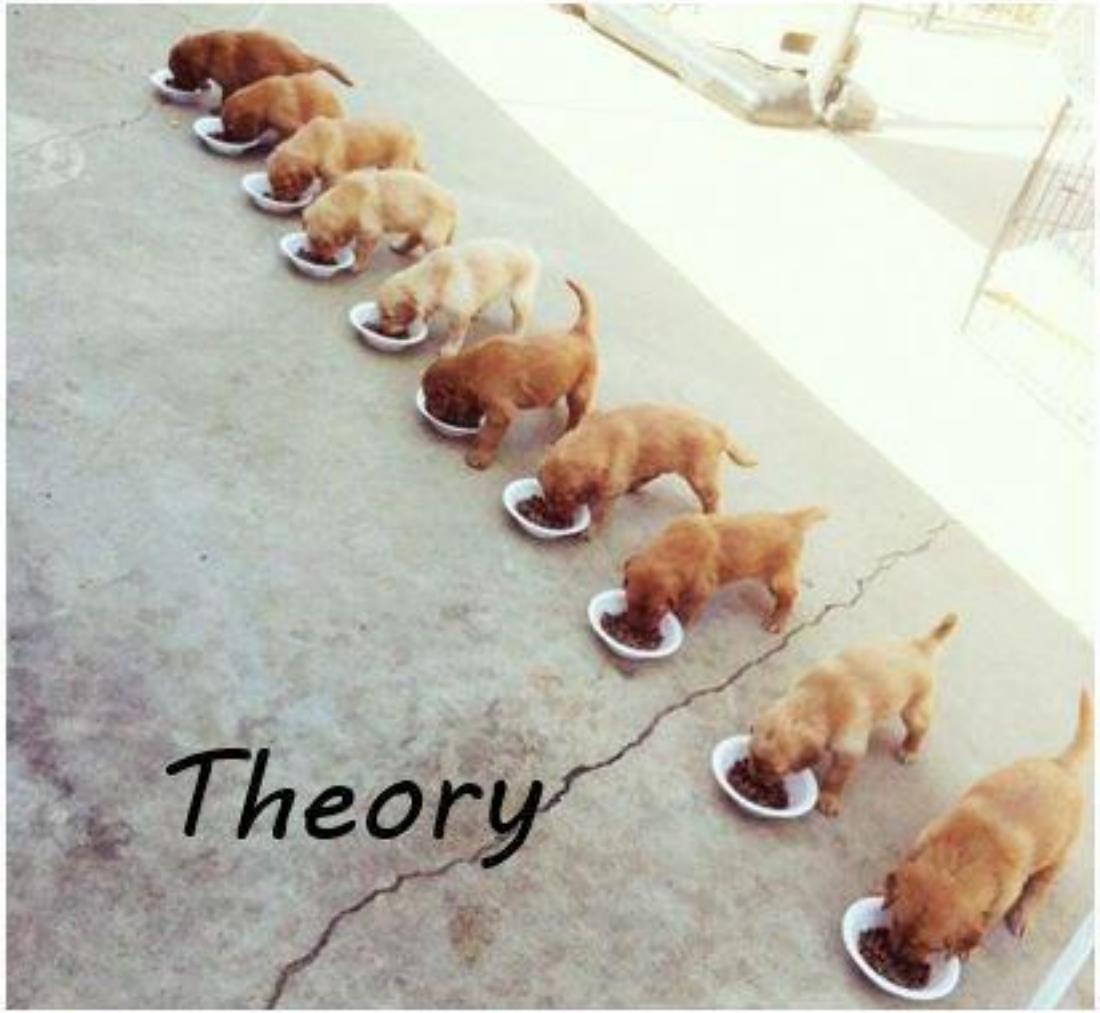
emscripten

Open Source LLVM to JavaScript compiler

@boyanio

```
emcc main.c -o main.html `
-s USE_PTHREADS=1 `
-s PTHREAD_POOL_SIZE=2
```

Multithreaded programming



@boyanio

https://hoonsbara.tistory.com/157

The future of Web belongs to those, who compile

Boyan Mihaylov / @boyanio / boyan.io