# WebAssembly is here. What does it mean for other Web frameworks?

Boyan Mihaylov
@boyanio
boyan.io

# 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!
```

# All I want from WebAssembly is…


# Performance

@boyanio

https://www.devproblems.com/laravel-speed-and-performance-optimization-101-the-guideline/

# WebAssembly provides consistent, predictable performance

16 Anim (2-214)

System WebAssembly

NumRunners 15

Close Controls

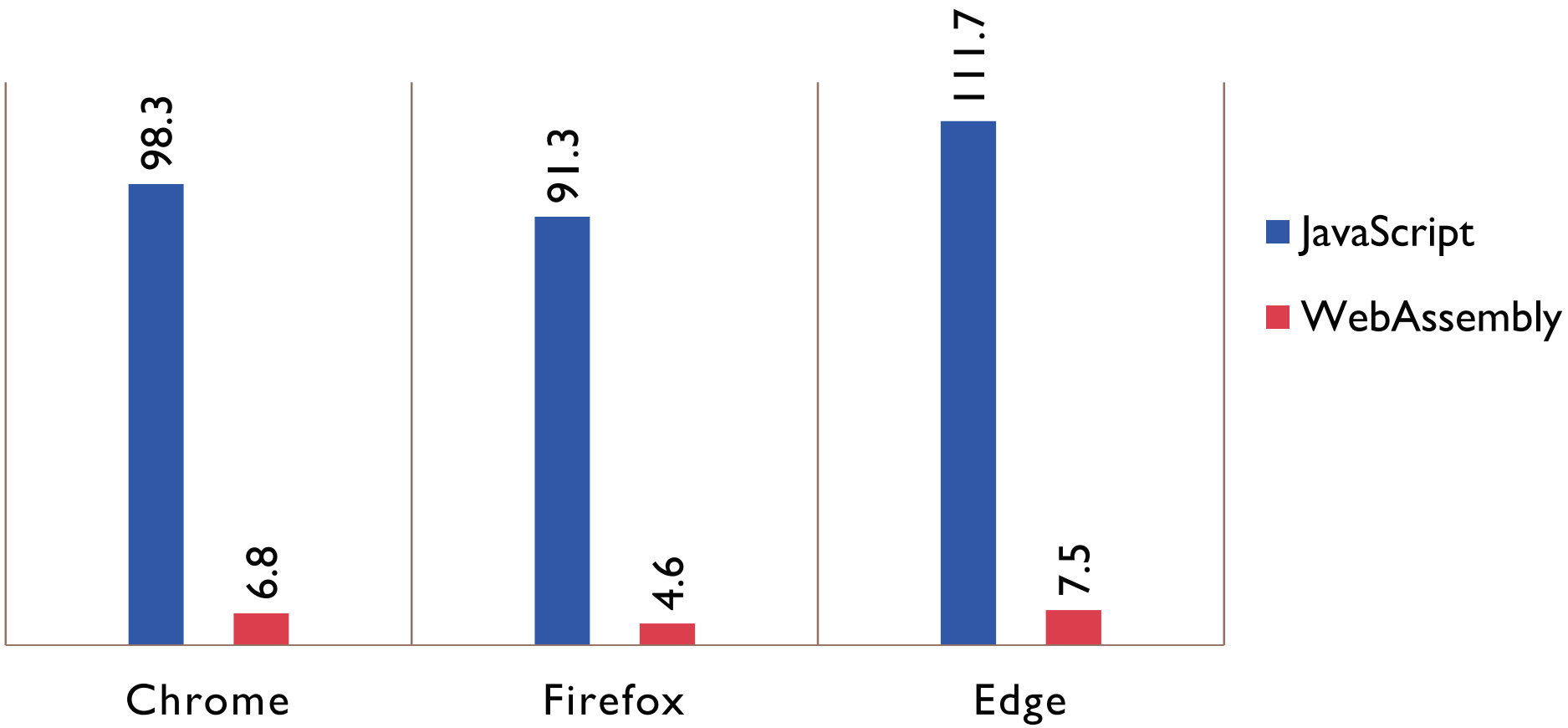
# 3D animation performance

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

@boyanio

# Performance comparison

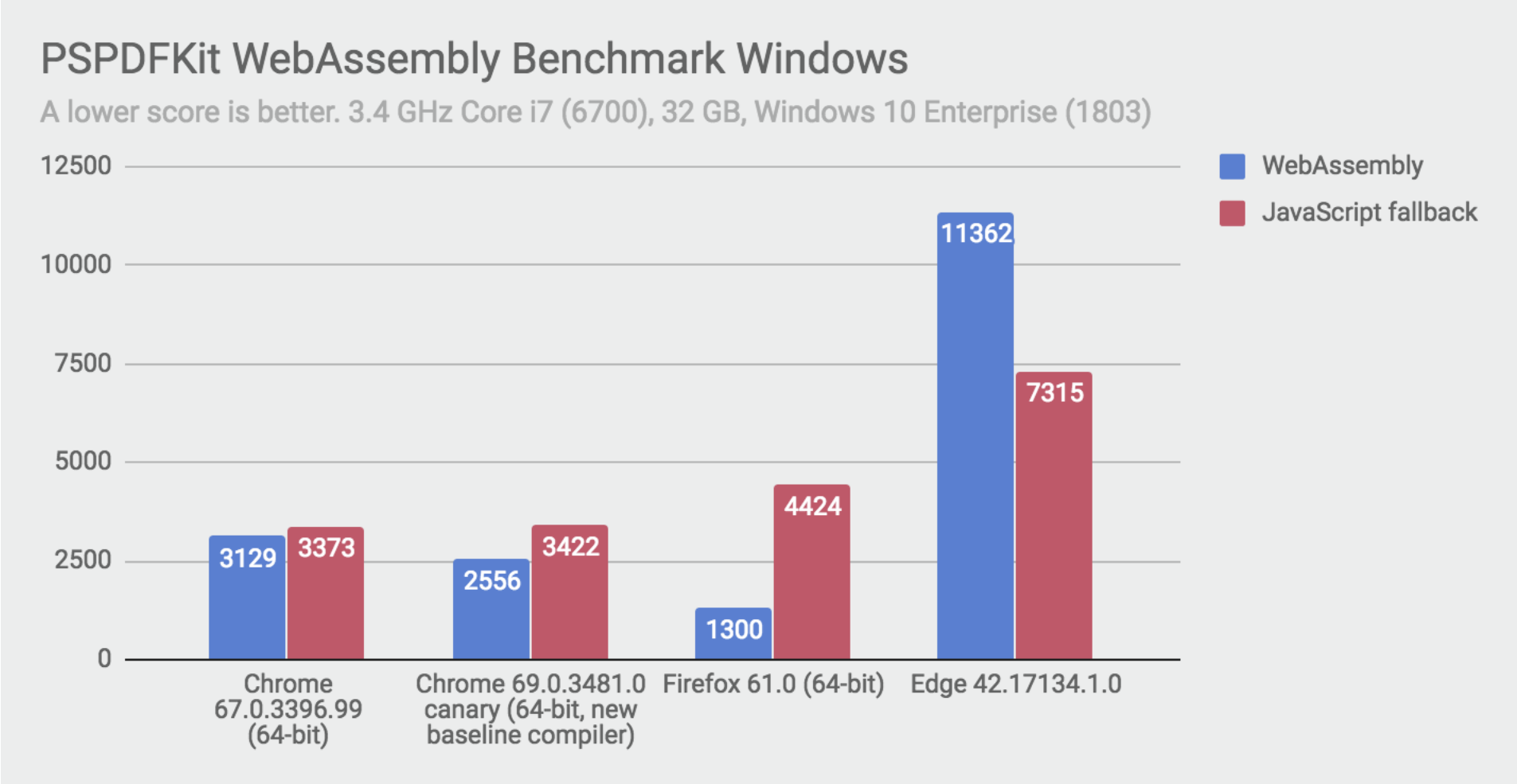
Average animation time (ms)





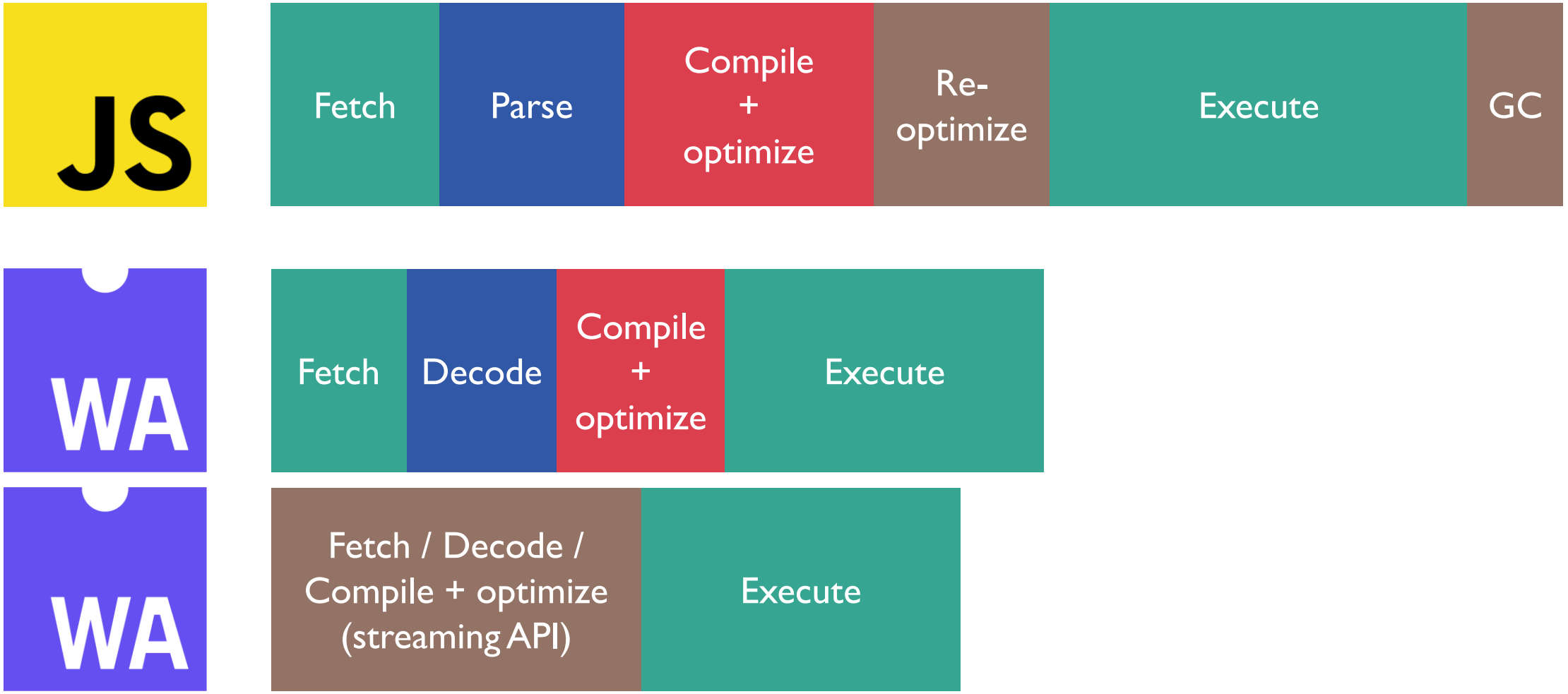

https://www.lucidchart.com/techblog/2017/05/16/webassembly-overview-so-fast-so-fun-sorta-difficult/

# A real-world WebAssembly benchmark






https://pspdfkit.com/blog/2018/a-real-world-webassembly-benchmark/


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

@boyanio

# Code reusability



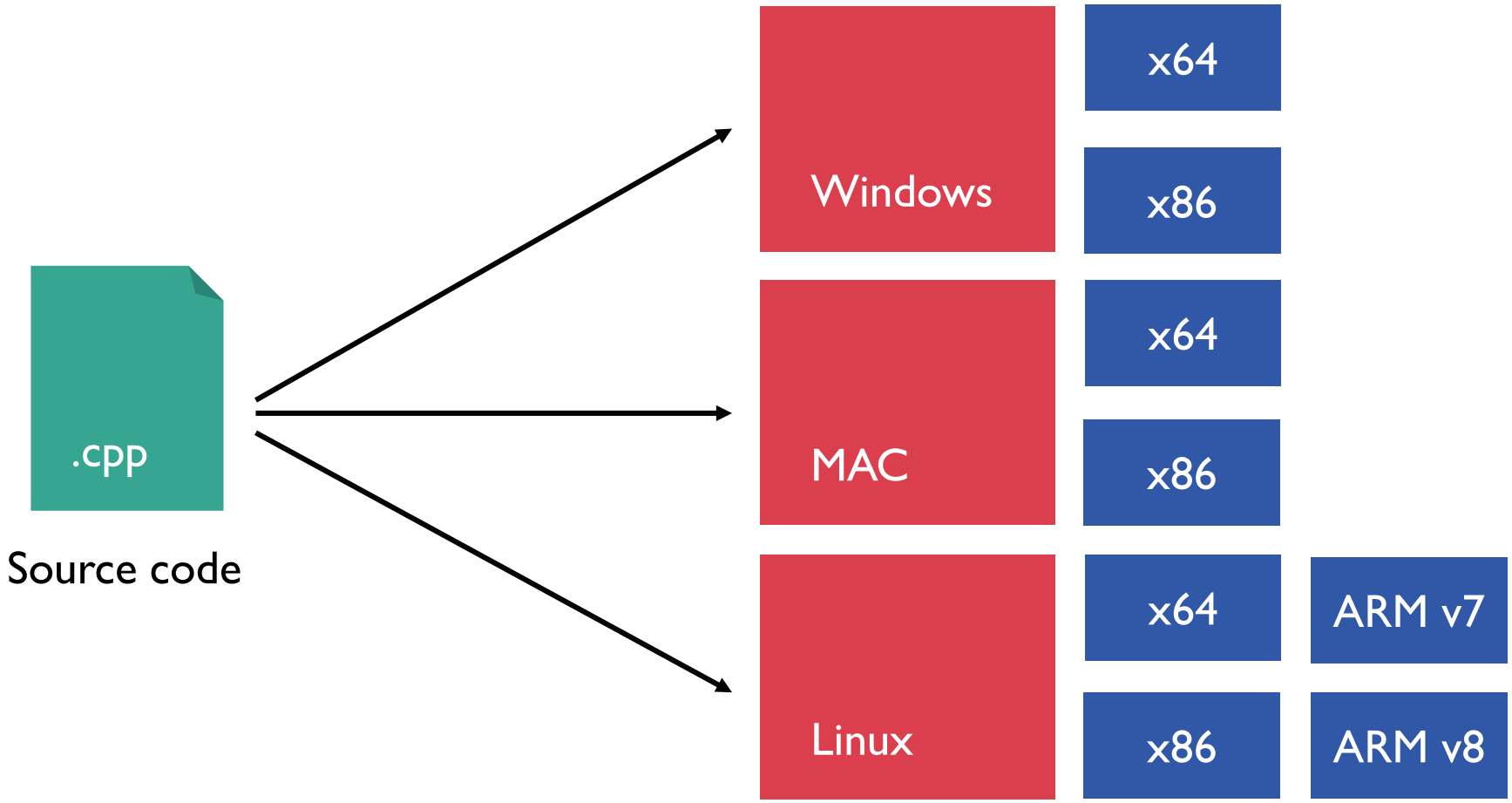
https://techcrunch.com/2016/07/05/lzlabs-launches-product-to-move-mainframe-cobol-code-to-linux-cloud/

The Adobe Flash plugin has crashed.
Send crash report

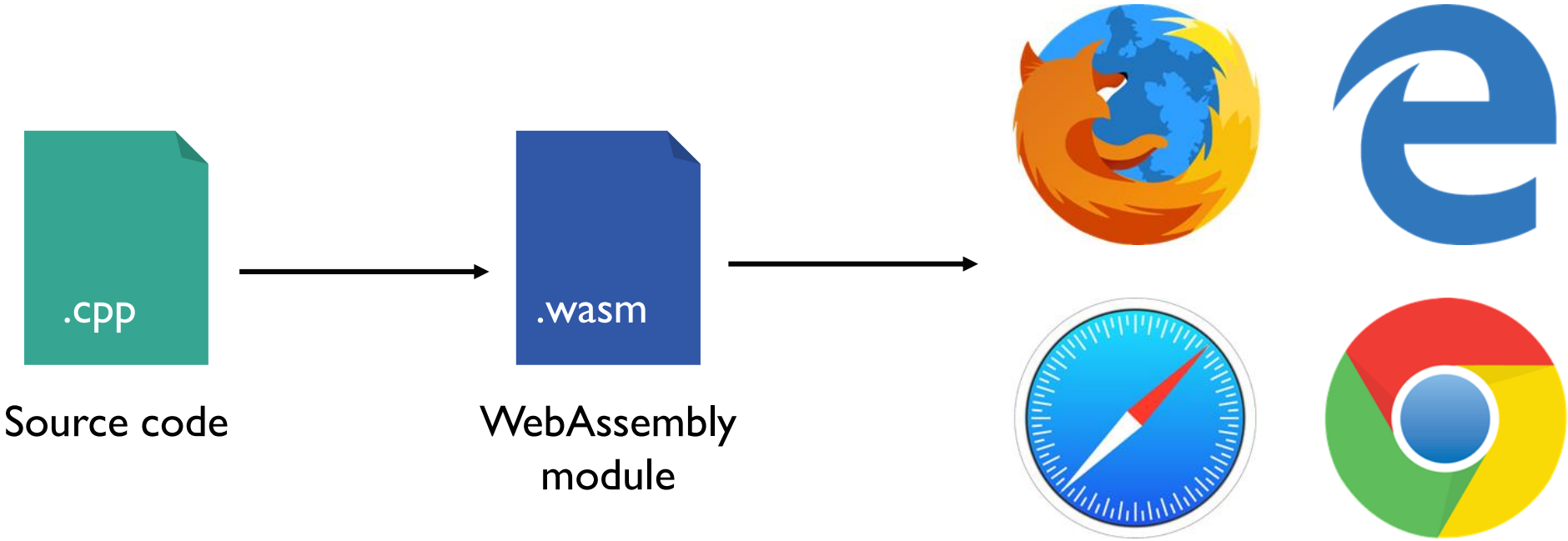
# Reusing code on the Web

@boyanio

# Traditional multi-target compilation



# Multi-target compilation with WebAssembly




@boyanio

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

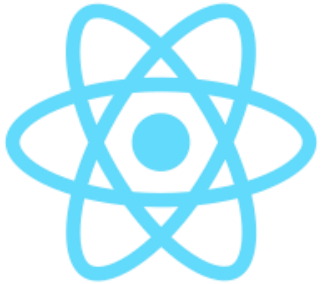



https://jeremybutterfield.files.wordpress.com/2014/12/conclusion.jpg


# WebAssembly will break the JavaScript monopoly

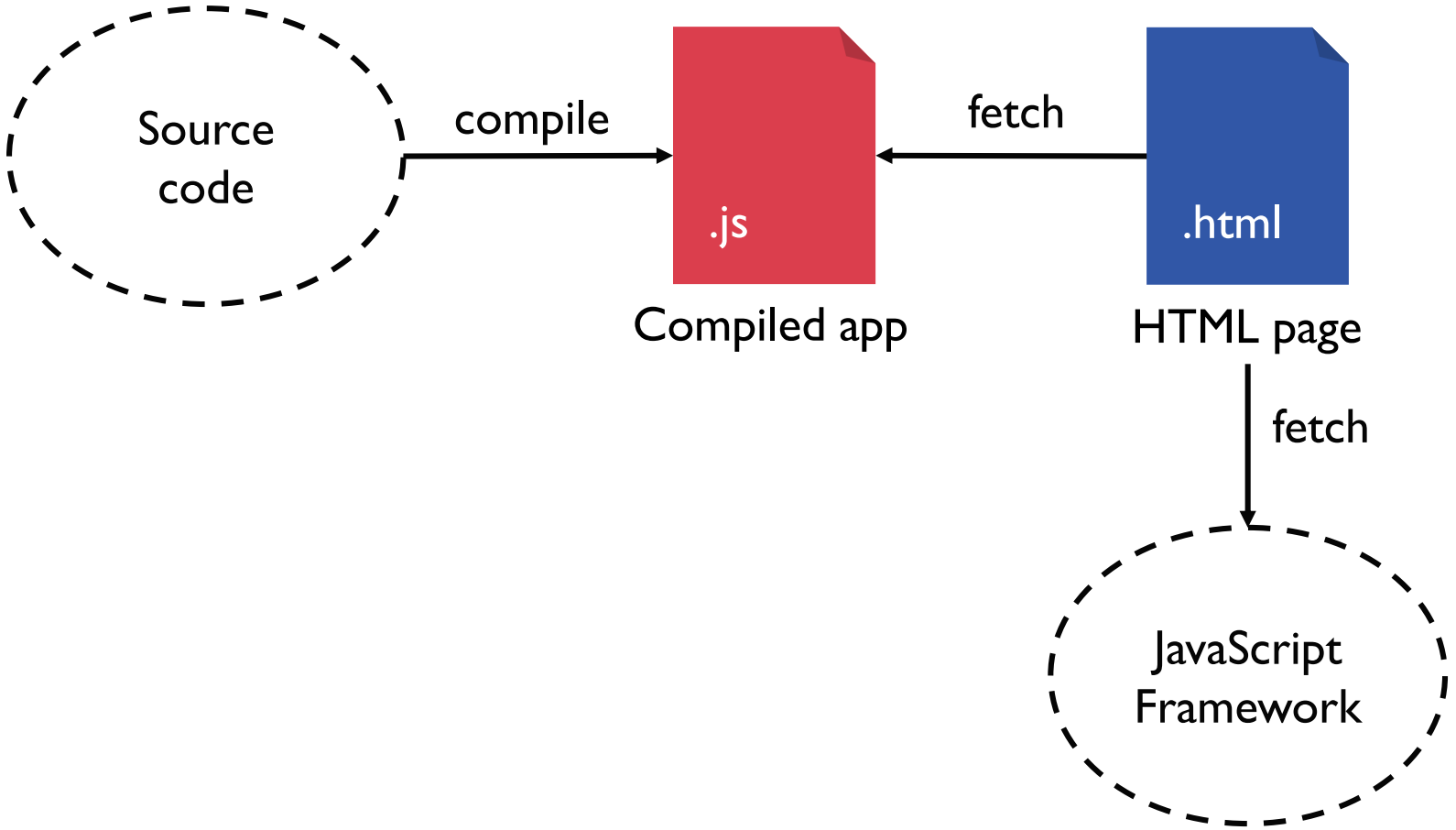


https://maas.museum/inside-the-collection/2010/03/17/see-you-round-like-a-monopoly-board/

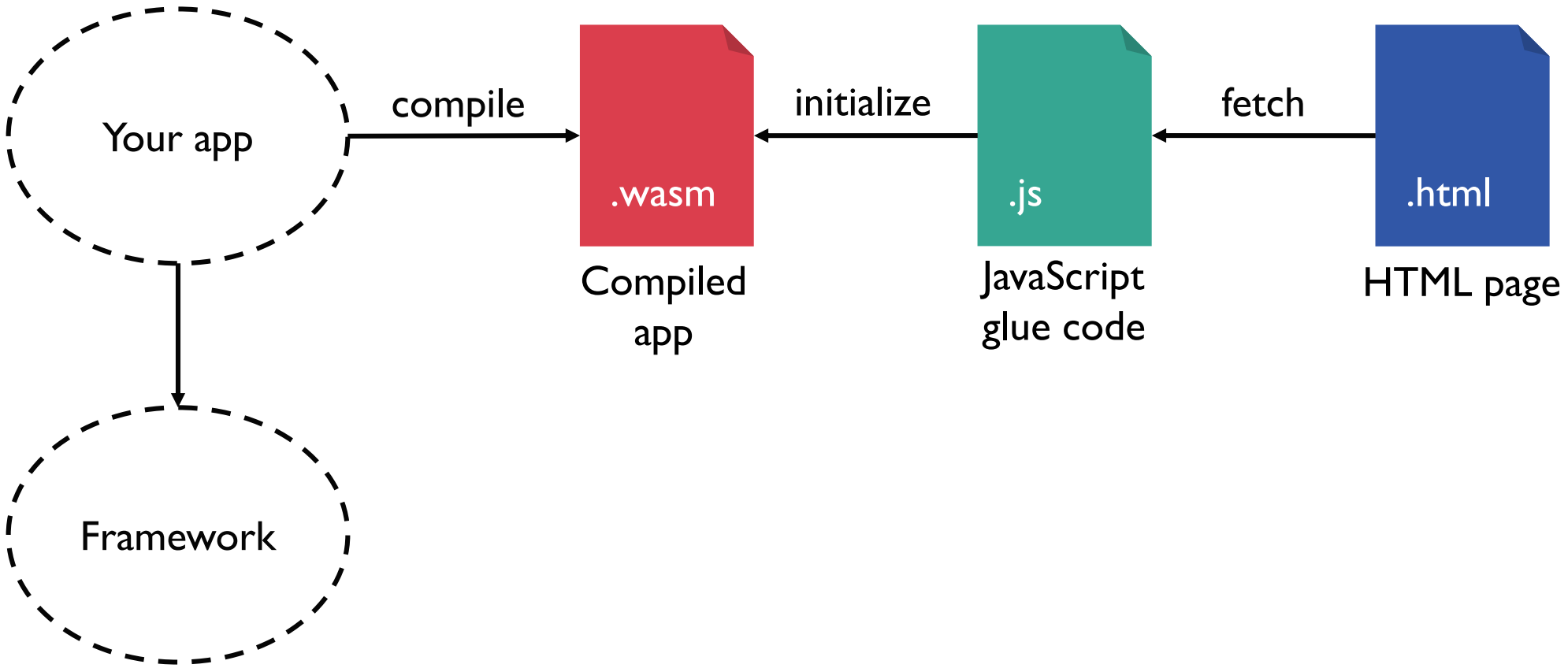

# The Web of JavaScript frameworks


# JavaScript frameworks architecture



# Rewriting existing JavaScript frameworks into a language that can be compiled to WebAssembly

# WebAssembly-compiled frameworks



# C++/Python → JavaScript → C++/Python

# No direct DOM access

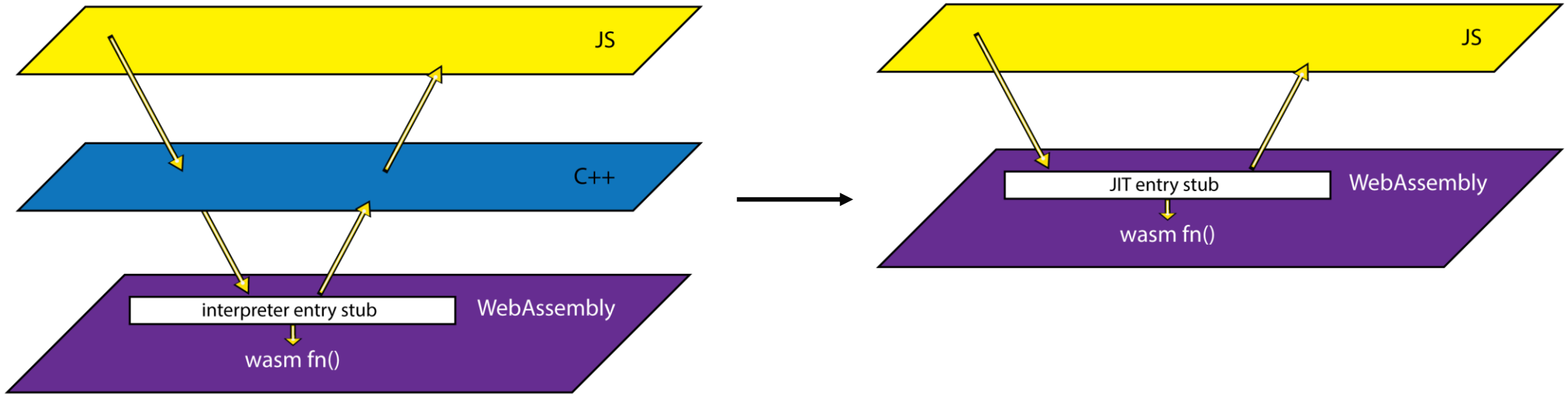
## index.c

```
extern void createElement(void);

int main(void)
{
  createElement();
  createElement();
  …
  return 0;
}
```

## main.js

```
const imports = {
  createElement: () => {
    document.createElement(‘div’);
  }
};

WebAssembly.instantiate(…, imports);
```

# JavaScript → WebAssembly overhead




Firefox

Firefox 60+



https://blog.benj.me/2018/07/04/mozilla-2018-faster-calls-and-anyref/

# Easier to create fast native mobile apps



Rewriting *parts* of existing JavaScript frameworks into a language that can be compiled to WebAssembly

2

# Languages

- Java: RxJava
- JavaScript: RxJS
- C#: Rx.NET
- C#(Unity): UniRx
- Scala: RxScala
- Clojure: RxClojure
- C++: RxCpp
- Lua: RxLua
- Ruby: Rx.rb
- Python: RxPY
- Go: RxGo
- Groovy: RxGroovy
- JRuby: RxJRuby
- Kotlin: RxKotlin
- Swift: RxSwift
- PHP: RxPHP
- Elixir: reaxive
- Dart: RxDart

Readme.md

# LibSass - Sass compiler written in C++

Currently maintained by Marcel Greter (@mgreter) and Michael Mifsud (@xzyfer)
Originally created by Aaron Leung (@akhleung) and Hampton Catlin (@hcatlin)

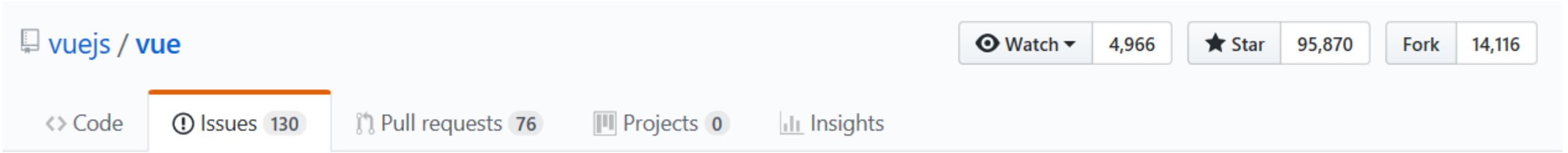

LibSass is just a library! If you want to use LibSass to compile Sass, you need an implementer. Some implementations are only bindings into other programming languages. But most also ship with a command line interface (CLI) you can use directly. There is also SassC, which is the official lightweight CLI tool built by the same people as LibSass.

## Excerpt of "sanctioned" implementations:

- https://github.com/sass/node-sass (Node.js)
- https://github.com/sass/perl-libsass (Perl)
- https://github.com/sass/libsass-python (Python)
- https://github.com/wellington/go-libsass (Go)
- https://github.com/sass/sassc-ruby (Ruby)
- https://github.com/sass/libsass-net (C#)
- https://github.com/medialize/sass.js (JS)
- https://github.com/bit3/jsass (Java)

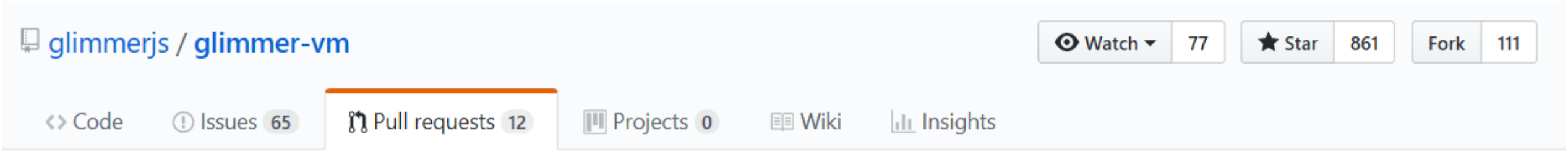
This list does not say anything about the quality of either the listed or not listed implementations!
The authors of the listed projects above are just known to work regularly together with LibSass developers.

@boyanio




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

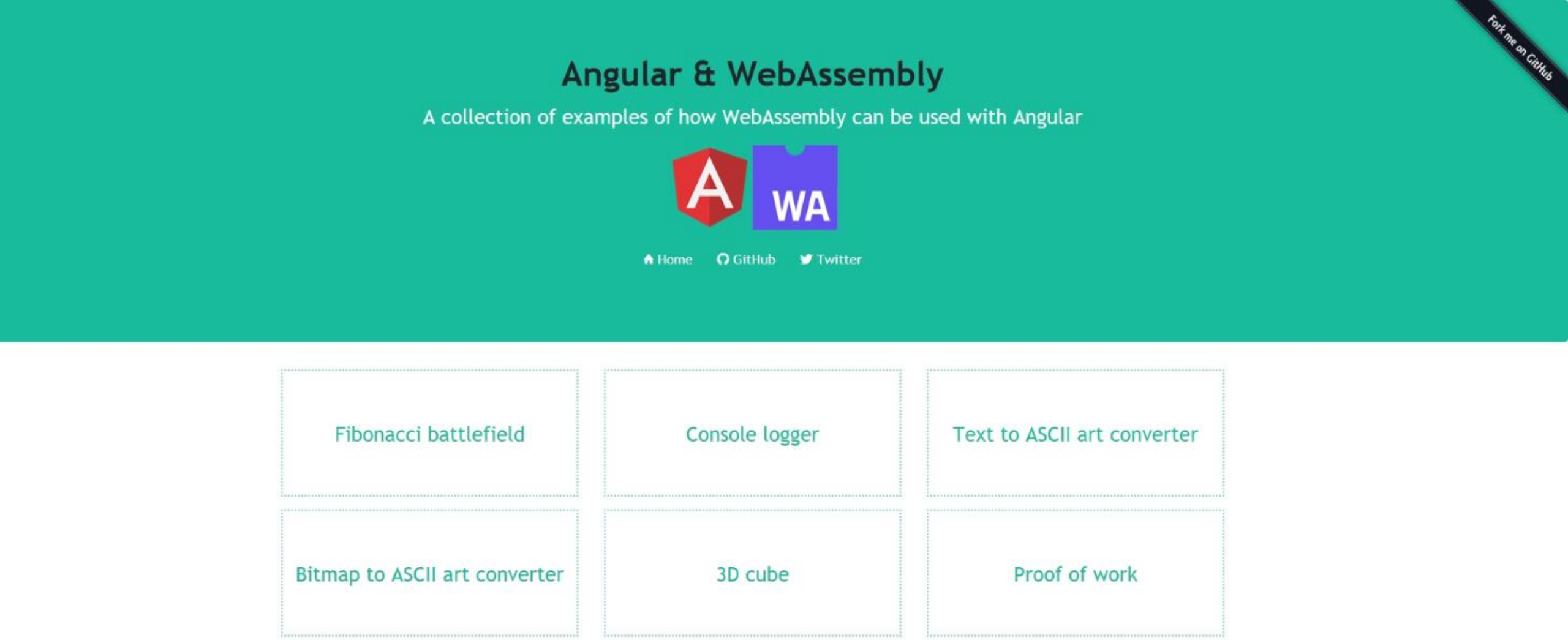
https://github.com/vuejs/vue/issues/8193




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

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

# Writing custom components in a language that can be compiled to WebAssembly

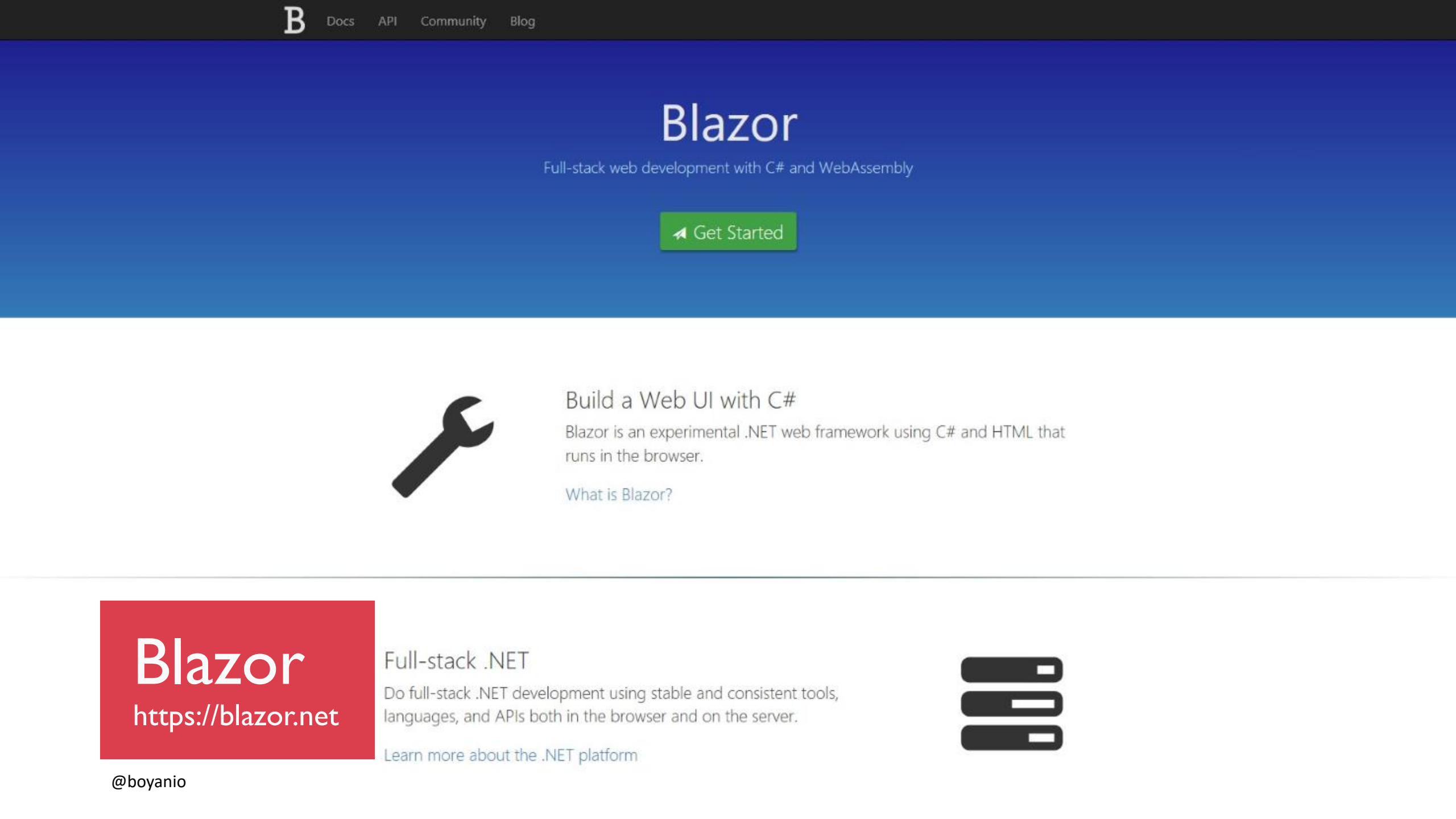



# Angular & WebAssembly

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

@boyanio

# The rise of non-JavaScript Web frameworks

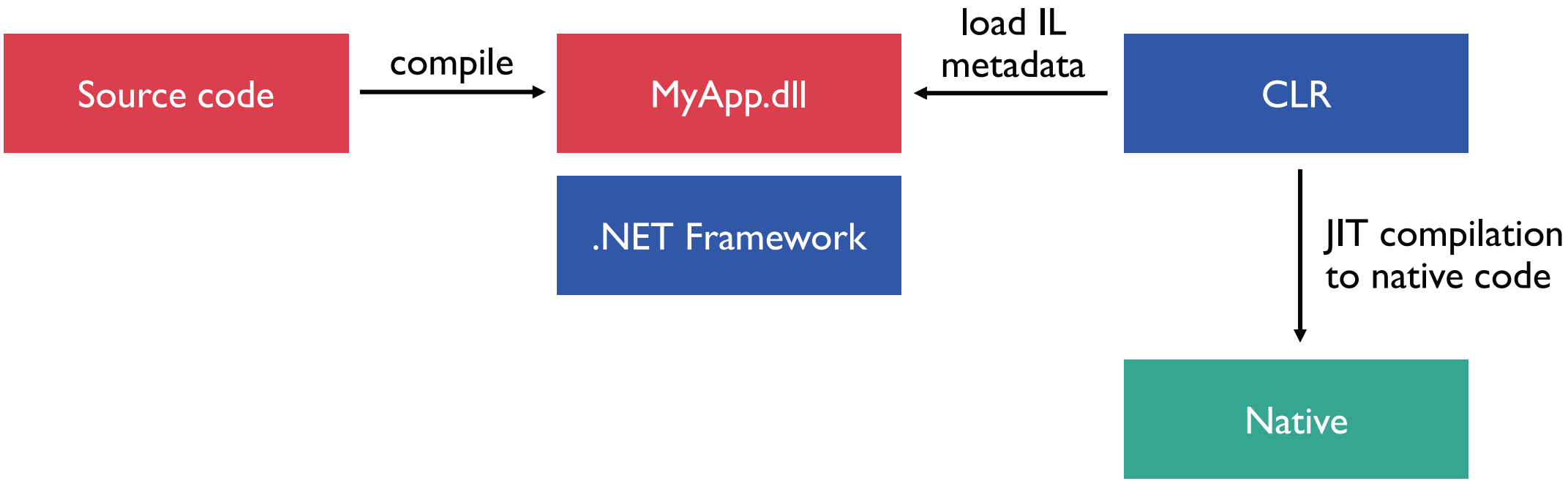



# Blazor

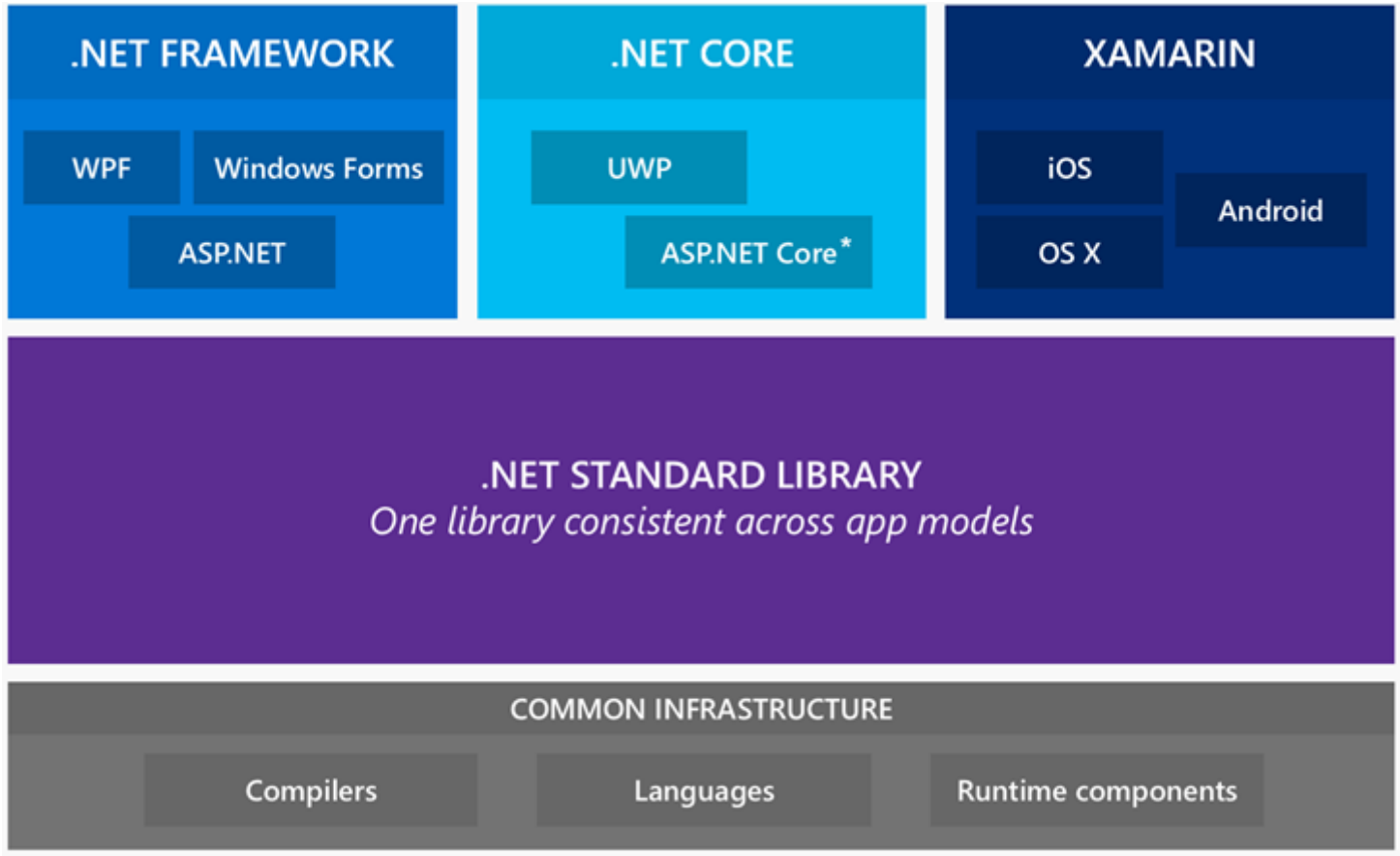
https://blazor.net

@boyanio

# Traditional .NET architecture



# .NET Core architecture

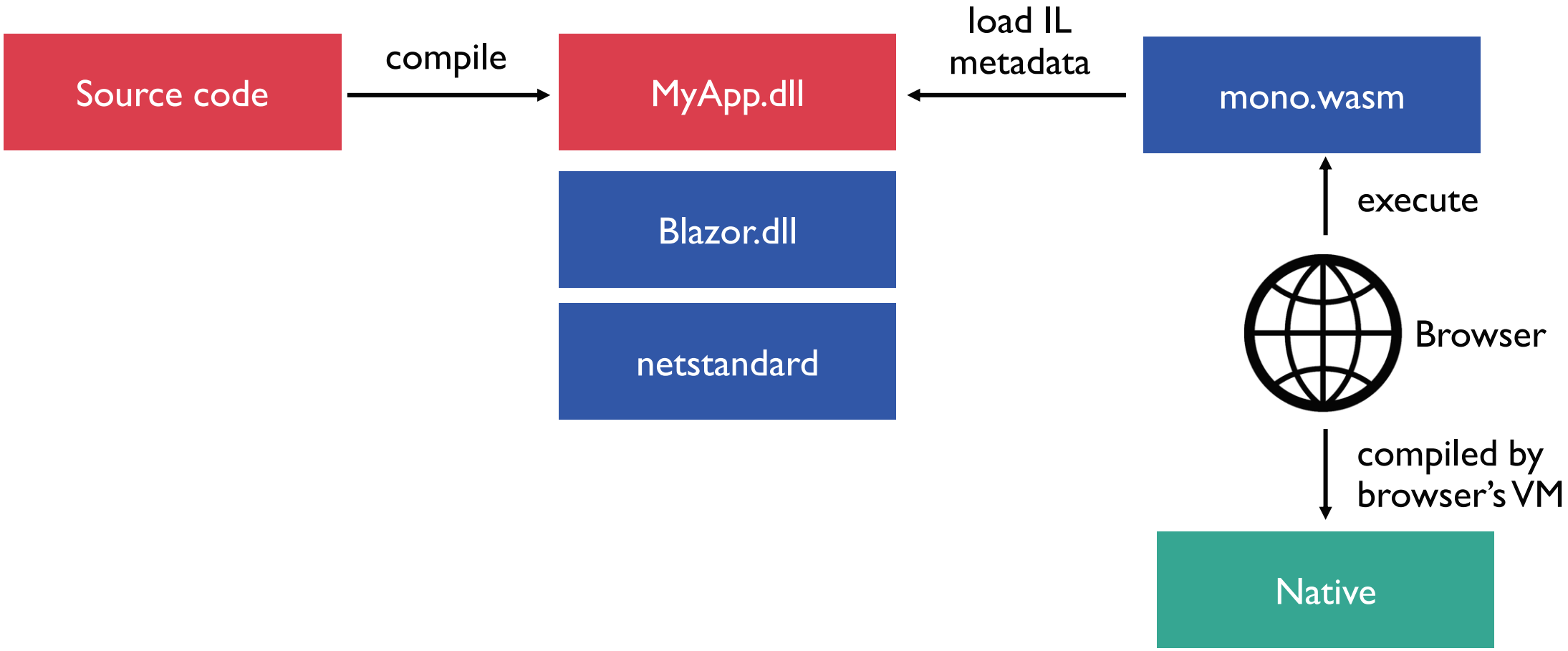


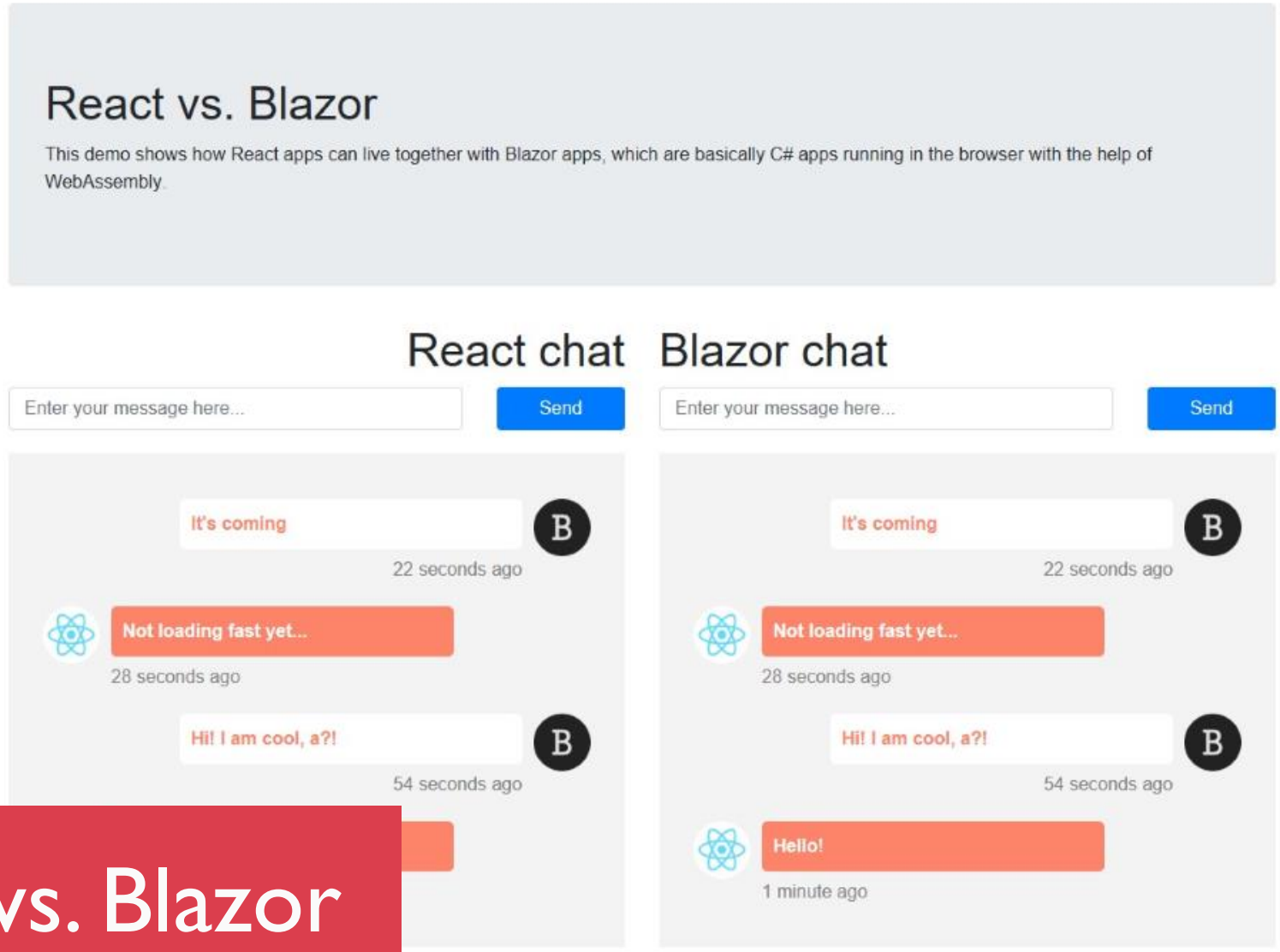



http://mode19.net/posts/dotnetcore/

Mono is an open source implementation of Microsoft's .NET Framework

# Blazor architecture

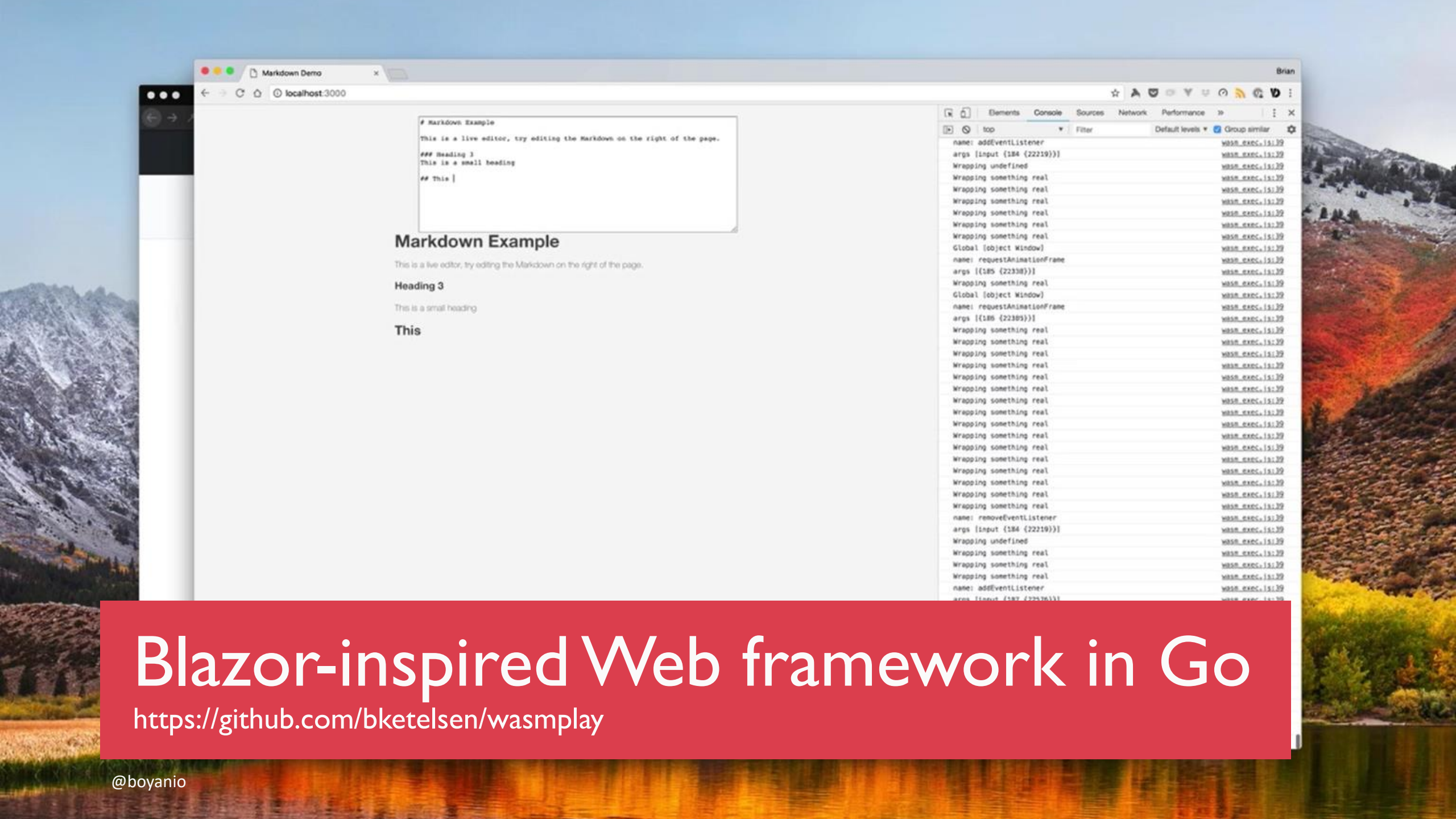





# React vs. Blazor

https://boyan.io/react-blazor/

@boyanio




# Blazor-inspired Web framework in Go

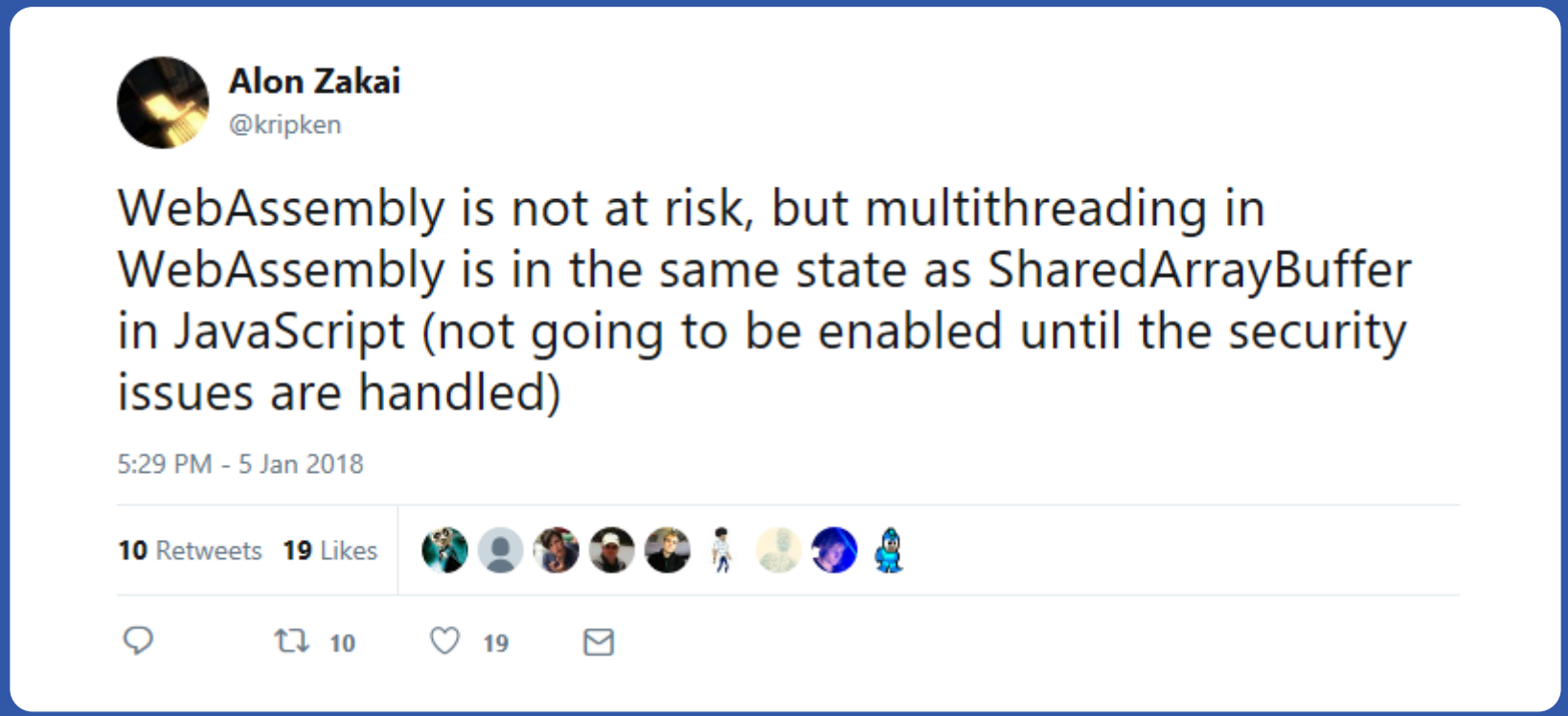
https://github.com/bketelsen/wasmplay

@boyanio

# WebAssembly enables different languages to work together on the Web

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

# 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

# The future of Web belongs to those, who compile

Boyan Mihaylov / @boyanio / boyan.io