WebAssembly

Grant Matejka

Tech Talk 2021-09-27

Agenda

I. Who Am I?
II. WebAssembly (aka wasm)?
    A. Background
    B. Goals
    C. What
    D. Why
    E. Where
    F. How
    G. Weaknesses
III. iFixit and wasm

`If a meeting doesn't have an agenda, they're probably winging it and wasting your time'

Who Am I?

Grant Matejka

Student Dev

Student

SE Undergrad
CS Masters

Hobbies

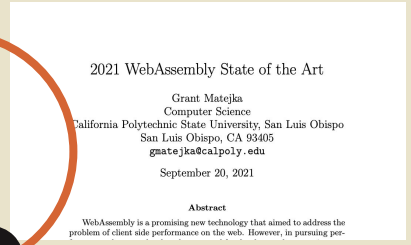
Reading & Running

Sharing my opinion with people who don't care

Being married

Thesis

WebAssembly
John Clements



2021 WebAssembly State of the Art

Grant Matejka
Computer Science
California Polytechnic State University, San Luis Obispo
San Luis Obispo, CA 93405
gmatejka@calpoly.edu

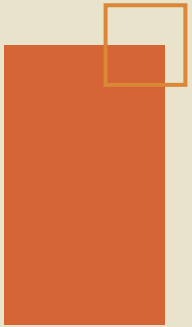
September 20, 2021

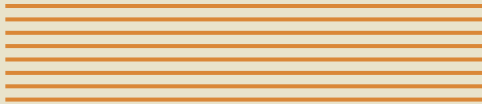
Abstract

WebAssembly is a promising new technology that aimed to address the problem of client side performance on the web. However, in pursuing per-

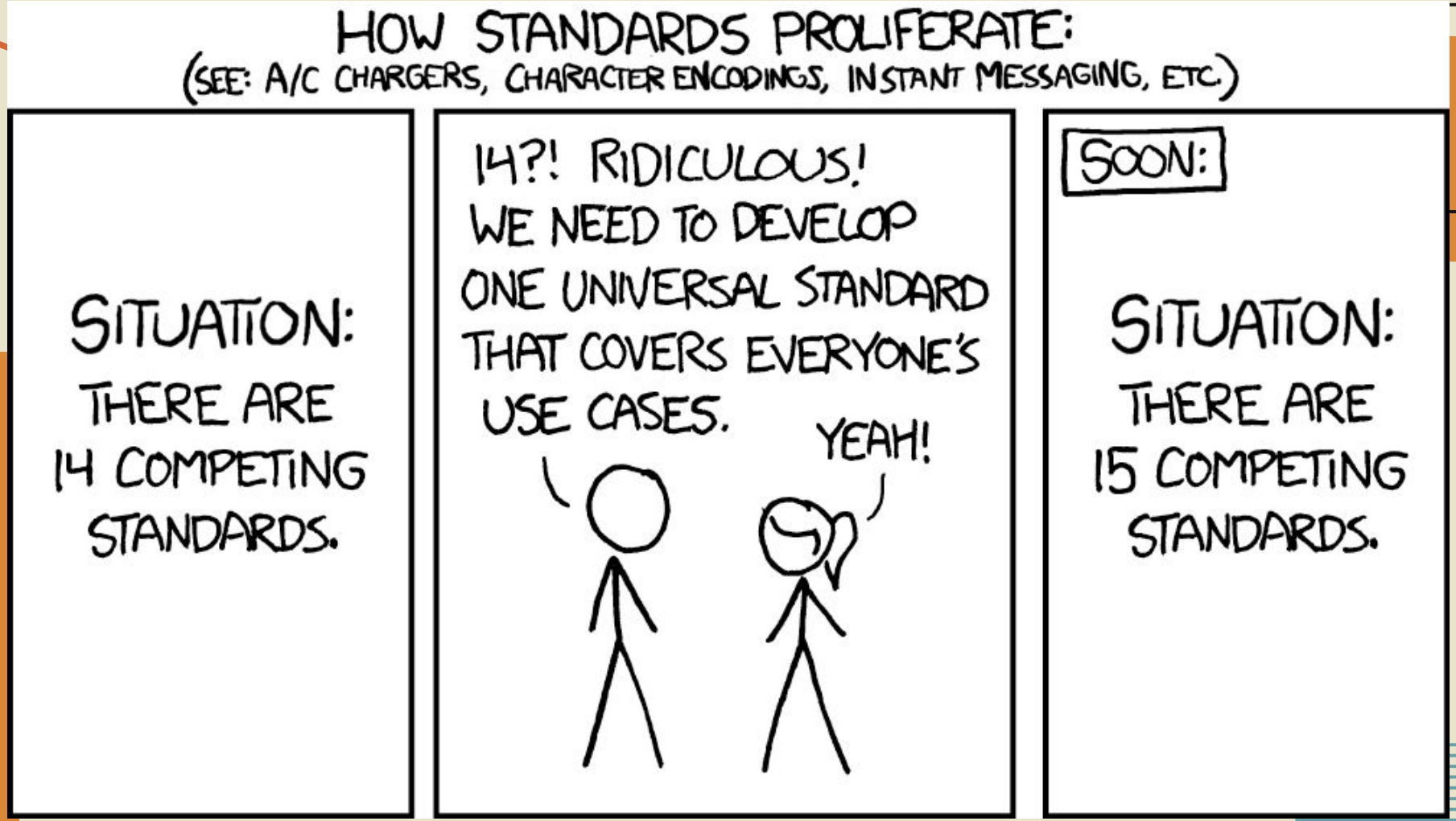
WebAssembly is...

A standardized byte code targeting a runtime and taking advantage of common hardware capabilities





Did someone say standard?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION:
THERE ARE
14 COMPETING
STANDARDS.
14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.
YEAH!
SOON:
SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

WebAssembly Did It

MVP implemented in major browsers 2017

Byte code for the web

- Consistent performance
  - Fast parse, execute
  - Size efficient
- Finally more than one language

We're not mad at js, just disappointed


WebAssembly - OTHER
WebAssembly or "wasm" is a new portable, size- and load-time-efficient format suitable for compilation to the web.
Usage
% of all users
Global
93.8%
Current aligned
Usage relative
Date relative
Filtered
All
IE
Edge
Firefox
Chrome
Safari
Opera
Safari on iOS
Opera Mini
Android Browser
Opera Mobile
2-46
12-14
47-51
4-50
10-37
15
52
51-56
3.1-10.1
38-43
3.2-10.3
6-10
16-92
53-91
57-92
11-14
44-78
11-14.7
2.1-4.4.4
12-12.1
11
93
92
93
14.1
79
15
all
93
64
93-94
94-96
15-TP

How Did We Get Here?

Performance on the web is a dated desire

Initial team worked on asm.js
- Subset of js focused on performance

They created a compiler for it: Emscripten
- c/c++ could now come to the browser

Performance was still lacking

WebAssembly was born

Goals of Wasm

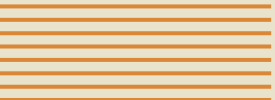
A technology is only as *good* as its values [Talk by Bryan Cantrill]

WebAssembly landing page says Wasm wants to be:

- Efficient and Fast
- Safe
- Open and Debuggable
- Part of the Open Web Platform

High Level Goals:

- Portable and efficient compilation target that executes near native speed by taking advantage of common hardware capabilities
- Incremental specification and implementation
- Integrate and execute well within the existing web platform
- Support non browser embeddings
- Make a great platform

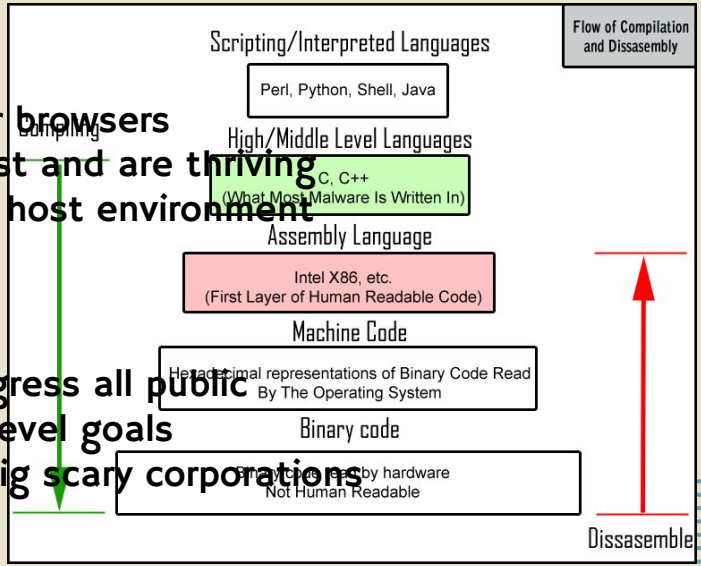


What is WebAssembly?

1. Byte Code (photo creds)
   a. Targeting simple & small but powerful instruction set
   b. Highly deterministic (AKA only working programs allowed), nondeterminism is limited and local
   c. Structured control flow
   d. Imported modules declare all types and functions at load time

2. Runtime
   a. Browser embeddings in all 4 major browsers
   b. Non browser/web embeddings exist and are thriving
   c. Memory safe and sandboxed from host environment

3. Community
   a. Very open source minded
   b. Standardization and proposal progress all public
   c. Built on a set of values and high level goals
   d. Collaboration even between the big scary corporations

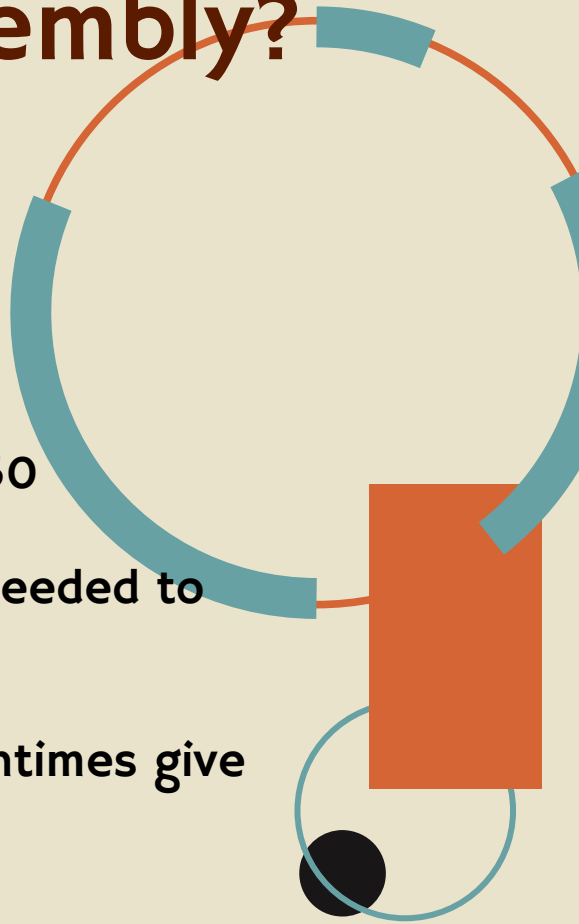


Why WebAssembly?

Wasm truly executes on and seeks out its values
It's easy to incorporate into existing projects
Consistent and safer performance than JavaScript [React's Pain]
- An option that is 'built for performance'

The runtime is awesome
- Sandboxed
- Memory safe
- Explicit communication in or out

Where do you use WebAssembly?



THE WEB
- Unity defaults to wasm for web projects
- ME's can now have macbooks

Cloud
- Shopify uses it to execute untrusted code
- Fastly provides Lucet and has startup speeds under 50 microseconds
- ``If WASM+WASI existed in 2008, we wouldn't have needed to created Docker.'' [tweet]

IoT
- looks to gain a lot of new flexibility as lightweight runtimes give a lot of hope

Any love for Sentry? They're hold the Flask

	SSVM 👀	Lucet/wasm-time	WAVM	V8	docker+na-tive
nop 0	0.003 👍	0.002 👍	0.024	0.056	0.849
cat-sync 0	0.007 👍	0.573	0.029 👍	0.06	0.826
nbody-c 50M	3.716 👍	4.611	3.753	3.408 👍	4.128
nbody-cpp 50M	3.759 👍	4.705	3.741 👍	3.962	3.944
fannkuch-redux-c 12	28.06 👍	53.104	28.477	29.285	24.459 👍
mandelbrot-c 15K	10.347 👍	28.97	12.072 👍	18.062	16.05
binary-trees-c 18	1.328 👍	2.91	1.612 👍	2.002	17.191

Sales pitch for SSVM, but still cool

source

How does one WebAssembly?

```rust
use wasm_bindgen::prelude::*;

#[wasm_bindgen]
extern {
    // This is a js defined function
    pub fn alert(s: &str);
}

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

```javascript
import * as wasm from "game-of-life";


wasm.greet();
```

Little bit cooler example

Weaknesses of WebAssembly?

MVP approach means that some features aren't quite there yet

- Questionable implementation support
- Multithreading is in mixed states
- Garbage Collection is blocked (Reference Types)

SIMD just finished a while back

Other efforts include module (dynamic) linking and tail calls

I'll fanboy when wasm adds _____ .

iFixit & WebAssembly

No use cases immediately stand out

I consider much of our content ugc and static
Our role is most likely going to be a consumer of wasm rather than producer

If anything, wasm could serve as a useful server-side technology for internal tools/toys/projects

?

Thank You

Please let me know your thoughts