

WebAssembly becoming the biggest platform

Sven Sauleau

2019

Sven Sauleau @svensauleau



The Web (previously)



Java: “write once, run anywhere”

Desktop apps.

Web pages.

Server.

Android.

Smart Card (SIM, credit card, ...).

The Web (today)

JavaScript

Desktop apps.

Web pages.

Server.

**JavaScript became
mainstream** on the web

Just think about
how many devices run JS.

Java

May 23, 1995

23 years ago

JavaScript

December 4, 1995

23 years ago

¹source: Wikipedia

**All good,
but suddenly...**

The `<blink>` tag
stopped working.

JavaScript, what happened?

Loading time

Fetching.

Parsing source.

Compiling + optimizing $\xrightarrow{\infty}$ reoptimizing.

Performance

Dynamic and untyped.

Complex runtime.

Managed memory.

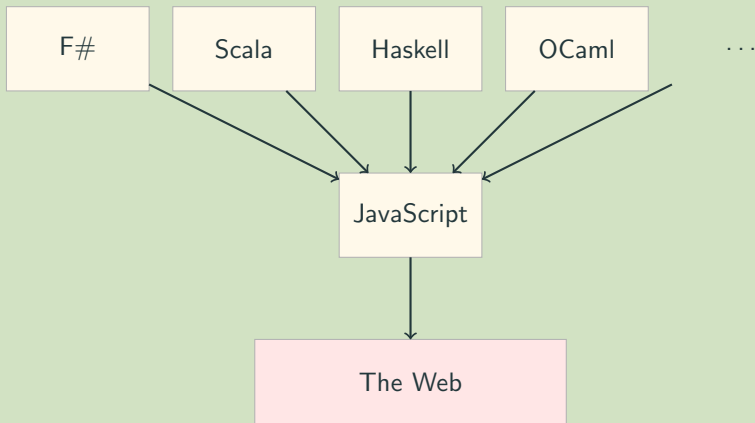
A few optimizations

Minification.

Tree shaking.

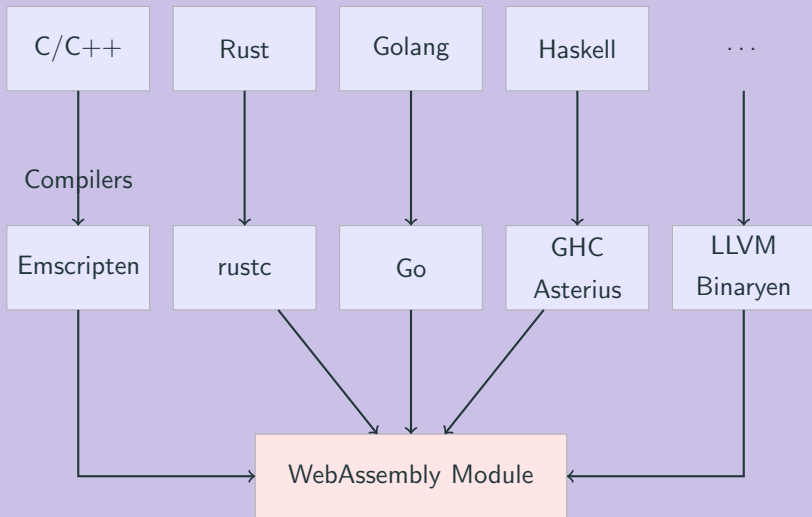
Static analysis is difficult.

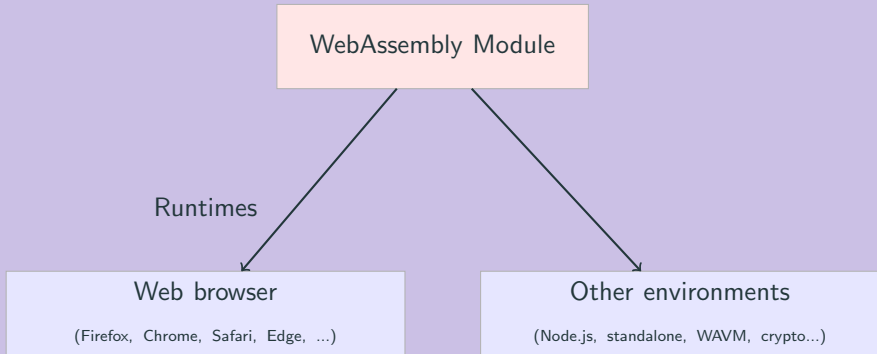
**Became a
compilation target**



WebAssembly, at the rescue?

Sources





WebAssembly is a
portable,
low-level,
safe
format.

Replace JavaScript with WebAssembly?

No!

JavaScript

Simple.

Accessible.

Easy to Debug and Test.

WebAssembly is designed to be a
complement to, not replacement of,
JavaScript.

Steps

`.wasm` $\xrightarrow{\text{decode}}$ `WebAssembly.Module`
 $\xrightarrow{\text{instantiate}}$ `WebAssembly.Instance`

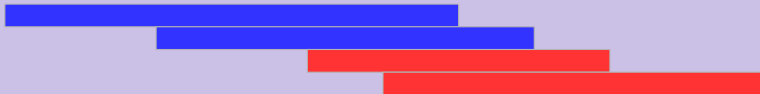
Efficient representation

Compact and easy to decode.

Streamable and parallelizable.

"Streamable"

Download → decode → instantiate → compile.



.wasm

```
00 61 73 6d 01 00 00 00 01 07 01 60  
02 7f 7f 01 7f 03 02 01 00 07 0a 01  
06 61 64 64 54 77 6f 00 00 0a 09 01  
07 00 20 00 20 01 6a 0b 00 19 04 6e  
61 6d 65 01 09 01 00 06 61 64 64 54  
77 6f 02 07 01 00 02 00 00 01 00 ...
```

WebAssembly Module

header

magic

version 1

type section

type #0

type #1

func section

func #1

func #2

code section

func #1 ...

func #2 ...

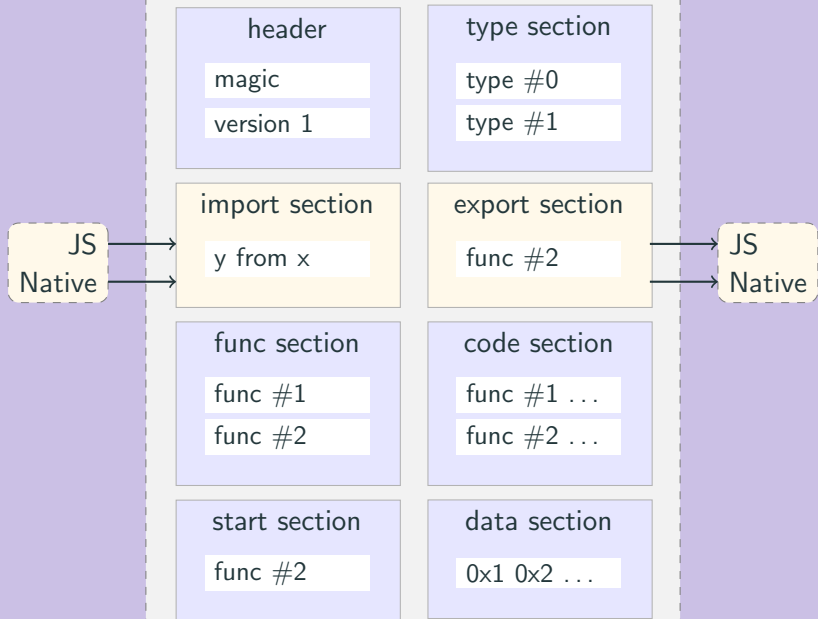
start section

func #2

data section

0x1 0x2 ...

WebAssembly Module



JS API

```
1  const buffer = ...;  
2  
3  const module = new WebAssembly.Module(buffer);  
4  const instance = new WebAssembly.Instance(  
5    module, importObject);  
6  
7  instance.exports.somefunc();
```

Under the hood

WebAssembly is a
stack-based
virtual machine

Register-based (x86, arm, ...)

```
1 mov %eax, 0x1
2 mov %rax, 0x1
3 add %eax, %rax
```

Stack-based (WebAssembly, call stack, ...)

```
1 i32.const 1
2 i32.const 1
3 i32.add
```

WebAssembly

Text
Format

module.wast:

```
1 (module
2   (func (export "addTwo") (param i32 i32) (result i32)
3     (get_local 0)
4     (get_local 1)
5     (i32.add)
6   )
7 )
```

Performance

WebAssembly is fast

Compiled to machine code.

Static analysis.

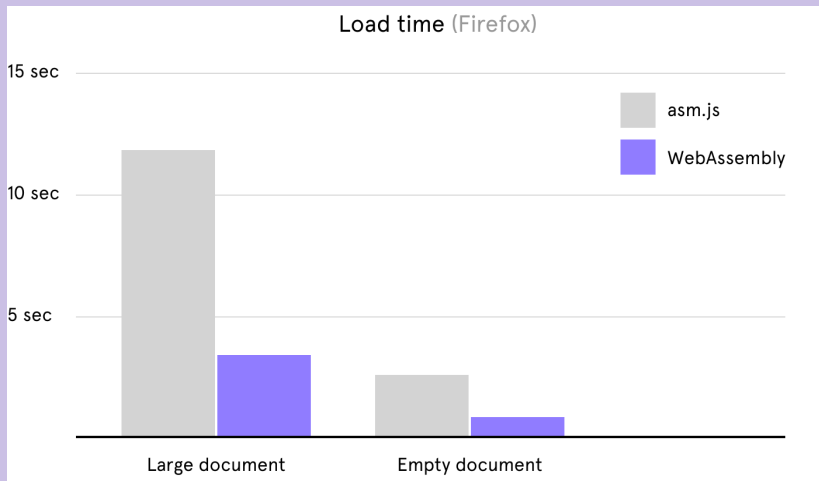
Optimized Ahead Of Time.

Crossing the boundary²

Can require a conversion.

Can require checks.

²<https://hacks.mozilla.org/2018/10/calls-between-javascript-and-webassembly-are-finally-fast-%F0%9F%8E%89/>



“Our load time improved by more than 3x [...]”

— Figma, medium

How to use it?

Likely:

```
1 $ compiler-x --target=wasm file
```

Languages³

- .Net
- Astro
- Brainfuck
- C / C# / C++
- Elixir
- Faust
- Forest
- Forth
- Haskell
- Golang
- Java
- Kotlin/Native
- Kou
- Lua
- OCaml
- Plorth
- Rust
- Turboscript
- Wah
- Wracket
- Xlang

³<https://github.com/appcypher/awesome-wasm-langs>

Browser support

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android
			49				
			63		10.3		
		58	64	11	11.2		
11	16	59	65	11.1	11.3	all	64
	17	60	66	TP			
	18	61	67				
			68				

Browser (no) support

WebAssembly → **JS compiler**

WebAssembly/binaryen

WebAssembly interpreter written in JavaScript

(WIP) xtuc/webassemblyjs

Other usages

Example Cloudflare Workers

Run code on edge on each request.

Links: [main.c](#), [blog](#) + [demo](#)

“**ewasm** is a restricted subset of WASM to be used for contracts in **Ethereum**.”

— ewasm

Unity and Unreal Engine use WebAssembly

Links: Funky Karts, AngryBots

What does it mean for JavaScript?

ES Module Integration ⁴

Import JS modules and values from Wasm.

Export JS module from Wasm.

⁴<https://github.com/WebAssembly/esm-integration>

with Webpack

module.c

```
1 #include <strings.h>
2 #include <webassembly.h>
3
4 EXPORT void test() {
5     console_log("Hi");
6 }
```

index.js

```
1 import("./module.c")
2   .then(({test}) => {
3     test();
4   });
```

JS-like languages

AssemblyScript: TypeScript $\rightarrow$ WebAssembly compiler⁵

```
1 export function add(a: i32, b: i32): i32 {  
2     return a + b;  
3 }
```

⁵[AssemblyScript.org](https://assemblyscript.org)

Work-In-Progress

WebAssembly:

```
1 (module
2   (func (export "fn") (param i64) (result i64)
3     (get_local 0)
4   )
5 )
```

JavaScript:

```
1 exports.fn(42n) === 42n
```

⁶<https://sauleau.com/notes/wasm-bigint.html>

⁷<https://github.com/WebAssembly/JS-BigInt-integration>

Integration with the host

Import Web APIs.

Manipulate JavaScript + DOM objects.

⁸<https://github.com/webassembly/host-bindings>

- Data structures
- Reference types
- Support more languages

⁹<https://github.com/webassembly/gc>

- Native threads
- Shared memory
- Atomics

¹⁰<https://github.com/webassembly/threads>

Demo

Thanks
