Web Assembly loves Ruby

Slides heavily influenced by **Yuta Saito**

Go watch his Ruby Kagi presentation

https://rubykaigi.org/2022/presentations/kateinoigakukun.html

Ruby (the good parts)

Designed to make programmers happy

Fast to write

Well-developed Gem Ecosystem

Ruby (the bad parts)

Some platforms can’t run Ruby easily

Some restricted platforms can’t install Ruby interpreter (e.g. Web browser, Mobile device)

Hard to run your Ruby program on user machine

Run on a server? Now you need to maintain the server

Ruby (the bad parts)

Installation
Making the first step easy is important for beginners

How often have you seen BUILD FAILED ?

WebAssembly is a game changer

A binary instruction format for a stack-based machine

Designed to be:

- Portable
- Language agnostic
- Size- and Load-time efficient
- Secure by Sandbox and more…

WebAssembly to the rescue

Some platforms can’t run Ruby easily
✅ Now the browser is everywhere

Installation battle
✅ Beginners can try Ruby on browser without any installation

Efficient and fast

The Wasm stack machine is designed to be encoded in a size- and load-time-efficient binary format. WebAssembly aims to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms.

Safe

WebAssembly describes a memory-safe, sandboxed execution environment that may even be implemented inside existing JavaScript virtual machines. When embedded in the web, WebAssembly will enforce the same-origin and permissions security policies of the browser.

Open and debuggable

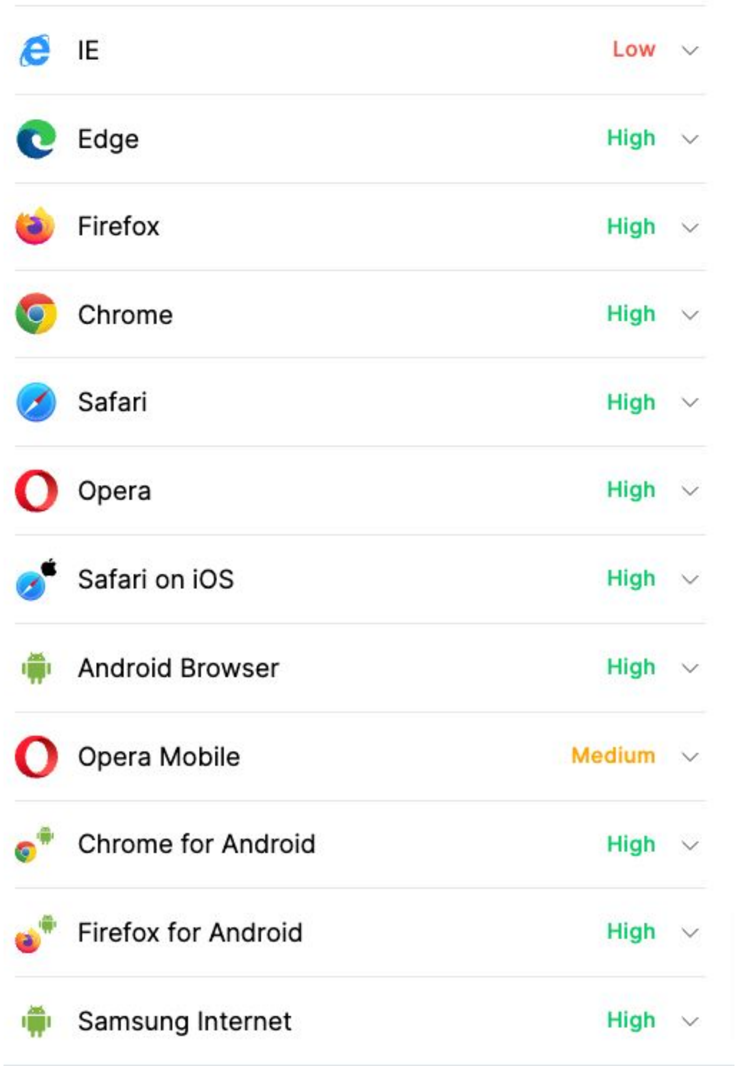
WebAssembly is designed to be pretty-printed in a textual format for debugging, testing, experimenting, optimizing, learning, teaching, and writing programs by hand. The textual format will be used when viewing the source of Wasm modules on the web.

Part of the open web platform

WebAssembly is designed to maintain the versionless, feature-tested, and backwards-compatible nature of the web. WebAssembly modules will be able to call into and out of the JavaScript context and access browser functionality through the same Web APIs accessible from JavaScript. WebAssembly also supports non-web embeddings.

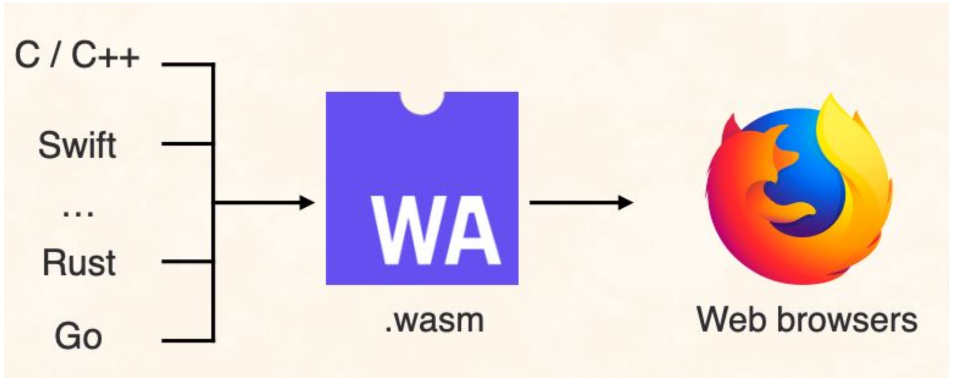
Support

We already have support on many of our favourite browsers

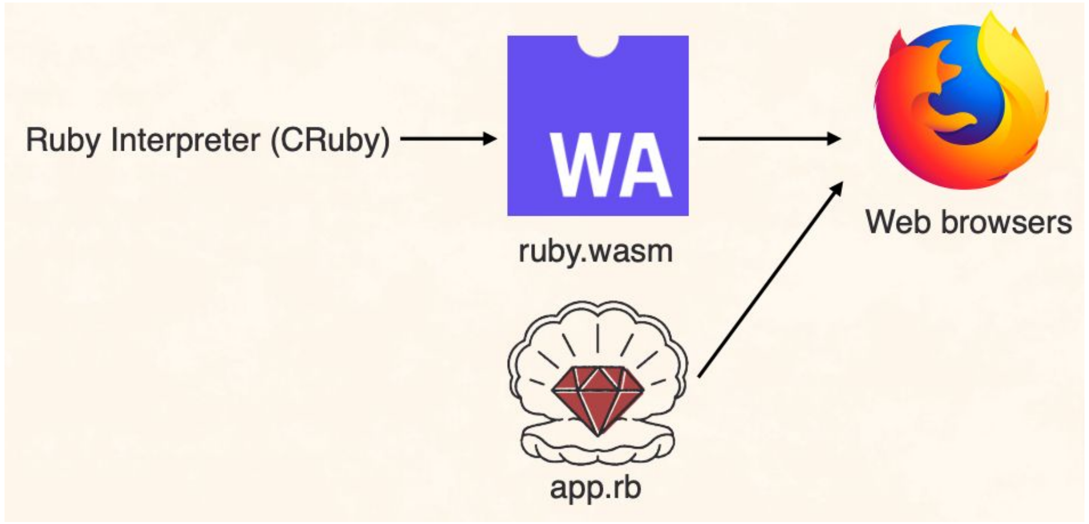


IE	Low
Edge	High
Firefox	High
Chrome	High
Safari	High
Opera	High
Safari on iOS	High
Android Browser	High
Opera Mobile	Medium
Chrome for Android	High
Firefox for Android	High
Samsung Internet	High

Execution flow of Web Assembly



Ruby on Web Assembly



WASI

The WebAssembly System Interface

WASI stands for WebAssembly System Interface. It's an API designed by the Wasmtime project that provides access to several operating-system-like features, including files and filesystems, Berkeley sockets, clocks, and random numbers

https://wasi.dev/

Demo Time

Ruby 3.2 will support Web Assembly and WASI out of the box

Limitations

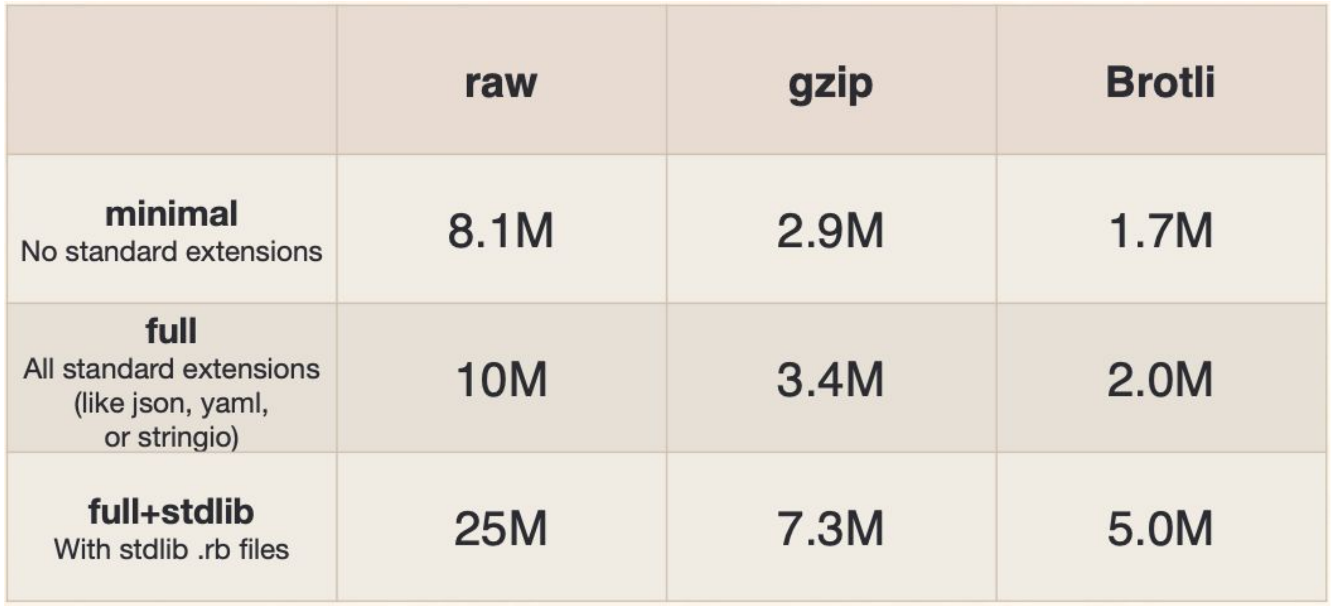
No Thread related API

Wasm and WASI don’t have thread spawning API yet

C Extension library must be linked statically

C libraries may have issues like nokogiri

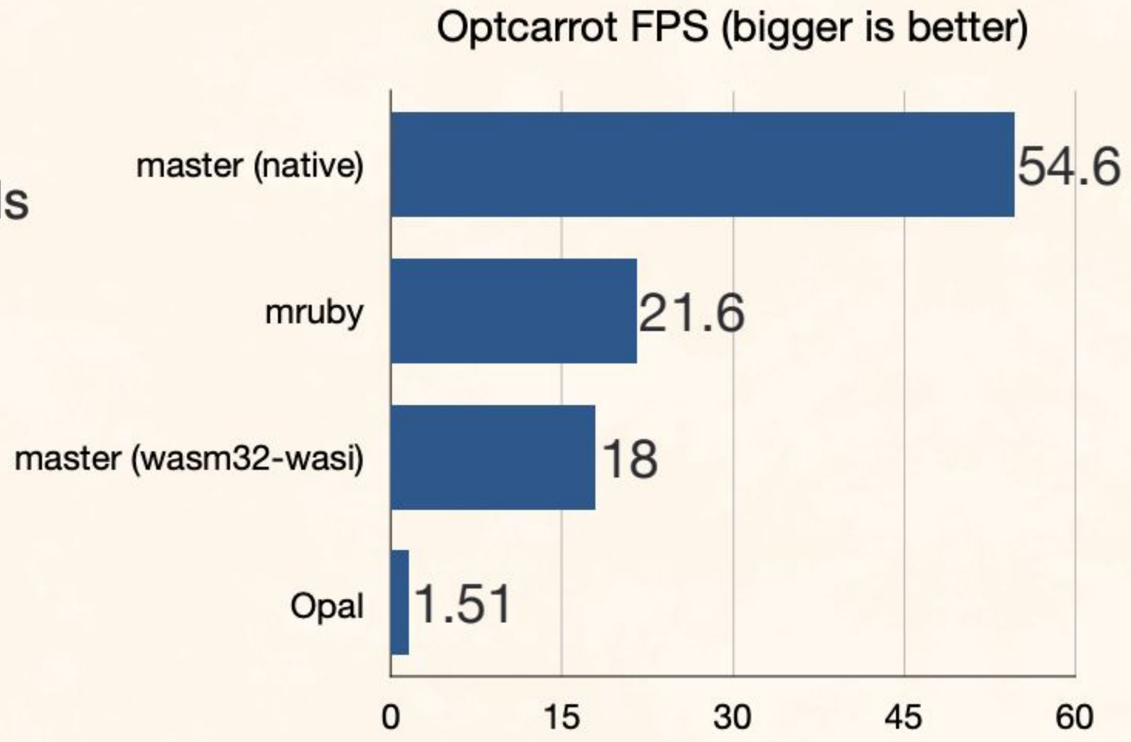
How big is the WASM binary?



	raw	gzip	Brotli
minimal No standard extensions	8.1M	2.9M	1.7M
full All standard extensions (like json, yaml, or stringio)	10M	3.4M	2.0M
full+stdlib With stdlib .rb files	25M	7.3M	5.0M

How fast is cruby on Web Asembly

- Near mruby speed
- Asyncify adds much overheads
- Environment
  - Node.js v17.9.0
  - Ubuntu 20.04
  - AMD Ryzen 9 5900HX



Recap

Ruby 3.2 will support WebAssembly and WASI

ruby.wasm provides npm packages and pre-compiled rubies

Try your favorite gems on https://irb-wasm.vercel.app/

Resources

https://github.com/ruby/ruby.wasm

https://webassembly.org

https://en.wikipedia.org/wiki/Ruby_MRI (cruby)

https://wasi.dev/ WASI

https://rubykaigi.org/2022/presentations/kateinoigakukun.html

https://irb-wasm.vercel.app/
https://github.com/mbasso/awesome-wasm
https://github.com/torch2424/wasmBoy