Ramón Huidobro

WebAssembly 💜 React

ramonh.dev

Containers? 🙋

WebAssembly? 🙋

I’m Ramón. (he/him)

From 🇨🇱, living in 🇦🇹

Co-Founder: BadWebsite.Club

DevRel Strategy Consultant

egghead Instructor

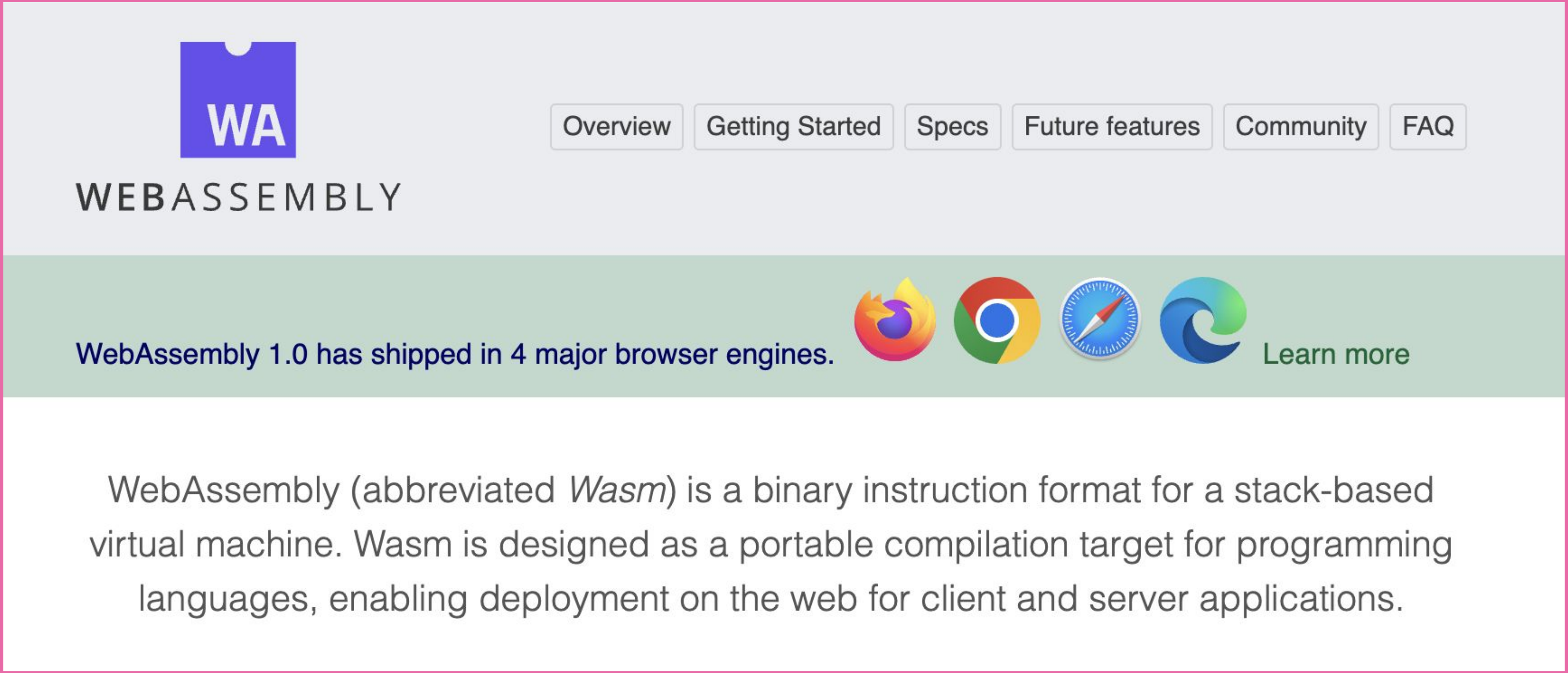
Community member

Mozilla tech speaker alum

Kids’ coding coach

Coding live streamer

What is WebAssembly (Wasm)?



WebAssembly 1.0 has shipped in 4 major browser engines.

WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.

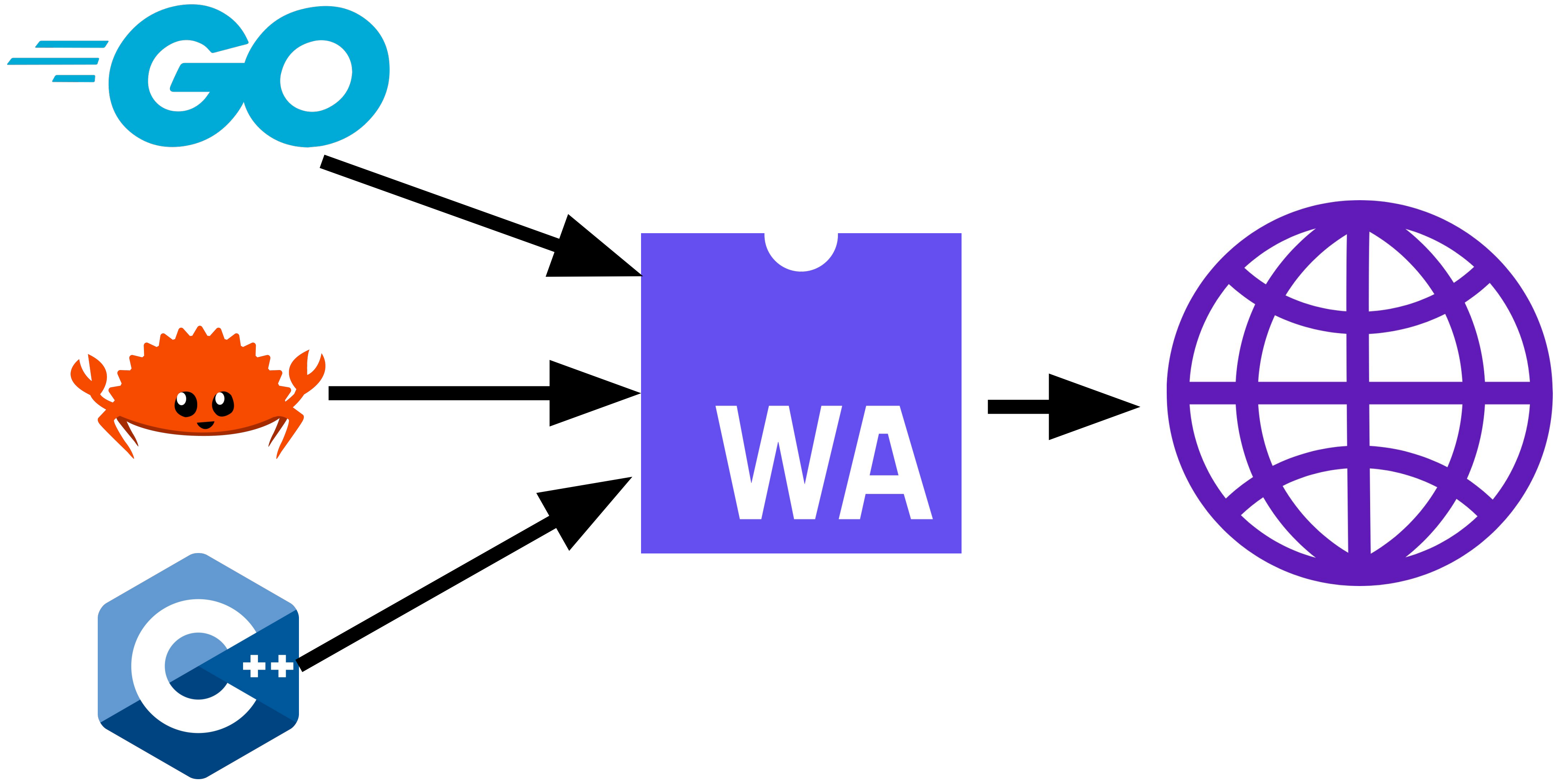
https://webassembly.org/

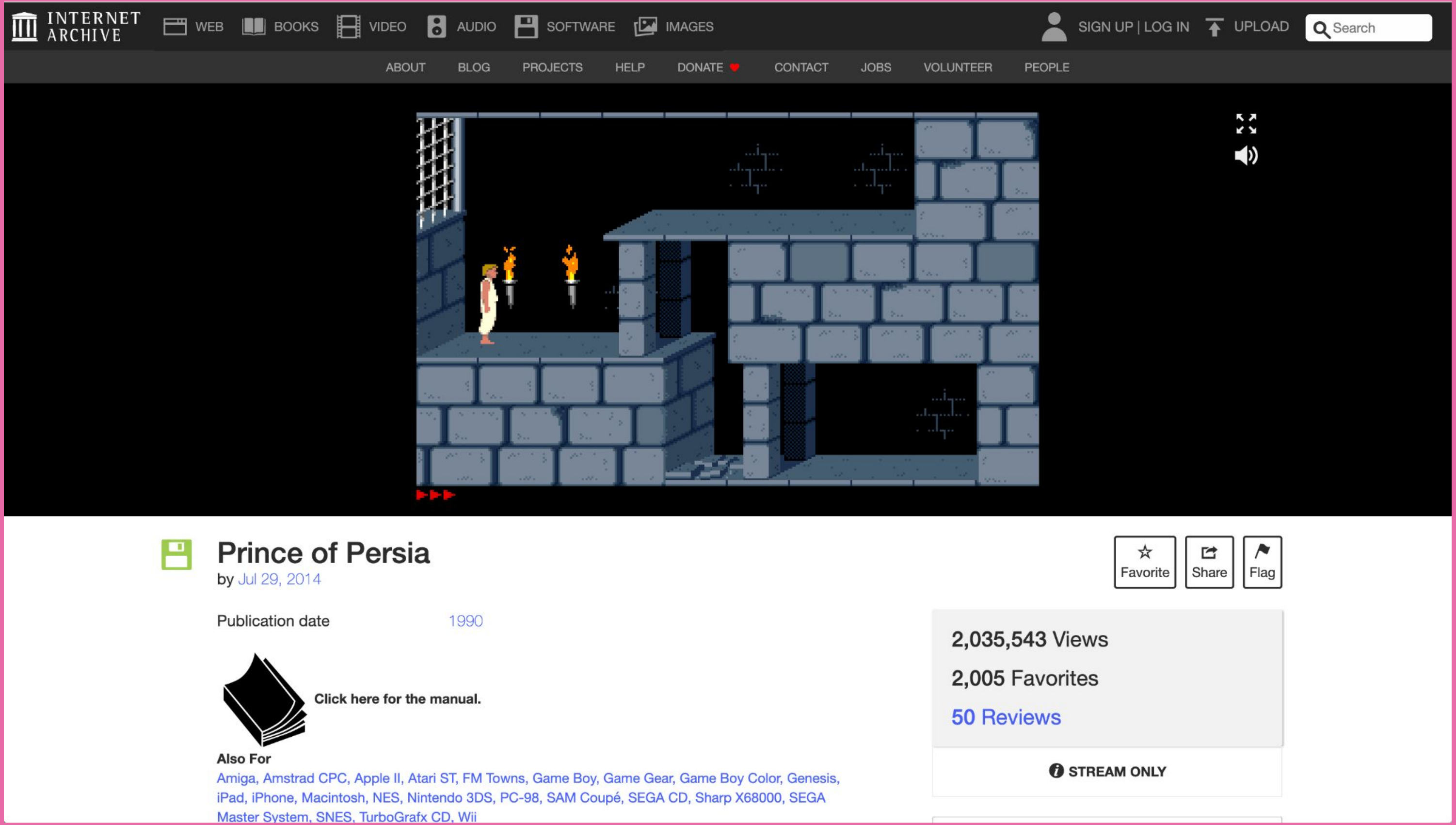
Wasm is <u>not</u> a programming language

Wasm is <u>not</u> a programming language

Wasm is a standard for low-level bytecode

Wasm is a <u>compilation target</u>


GO
WA



Prince of Persia
by Jul 29, 2014

Publication date 1990

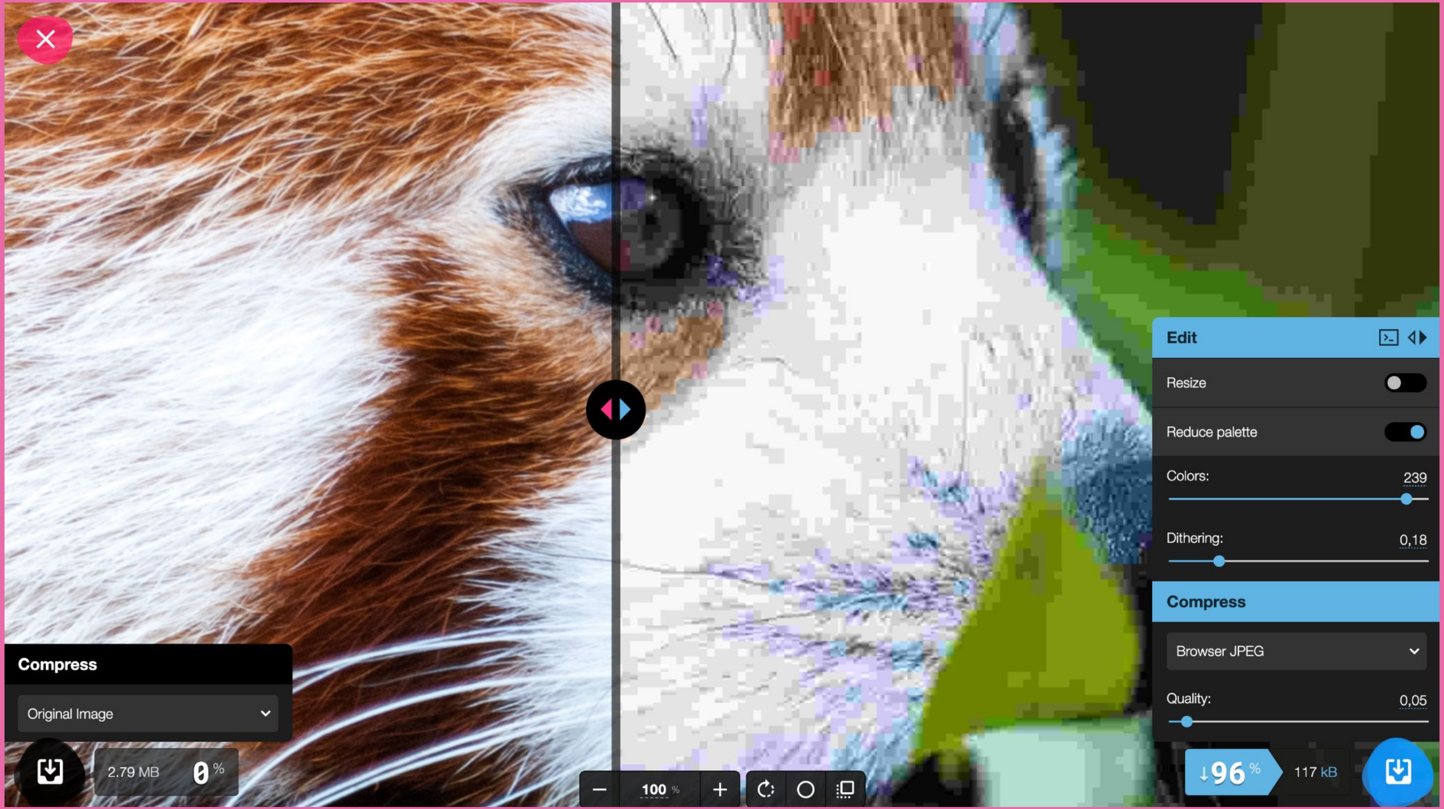
Click here for the manual.

Also For
Amiga, Amstrad CPC, Apple II, Atari ST, FM Towns, Game Boy, Game Gear, Game Boy Color, Genesis, iPad, iPhone, Macintosh, NES, Nintendo 3DS, PC-98, SAM Coupé, SEGA CD, Sharp X68000, SEGA Master System, SNES, TurboGrafx CD, Wii

2,035,543 Views
2,005 Favorites
50 Reviews

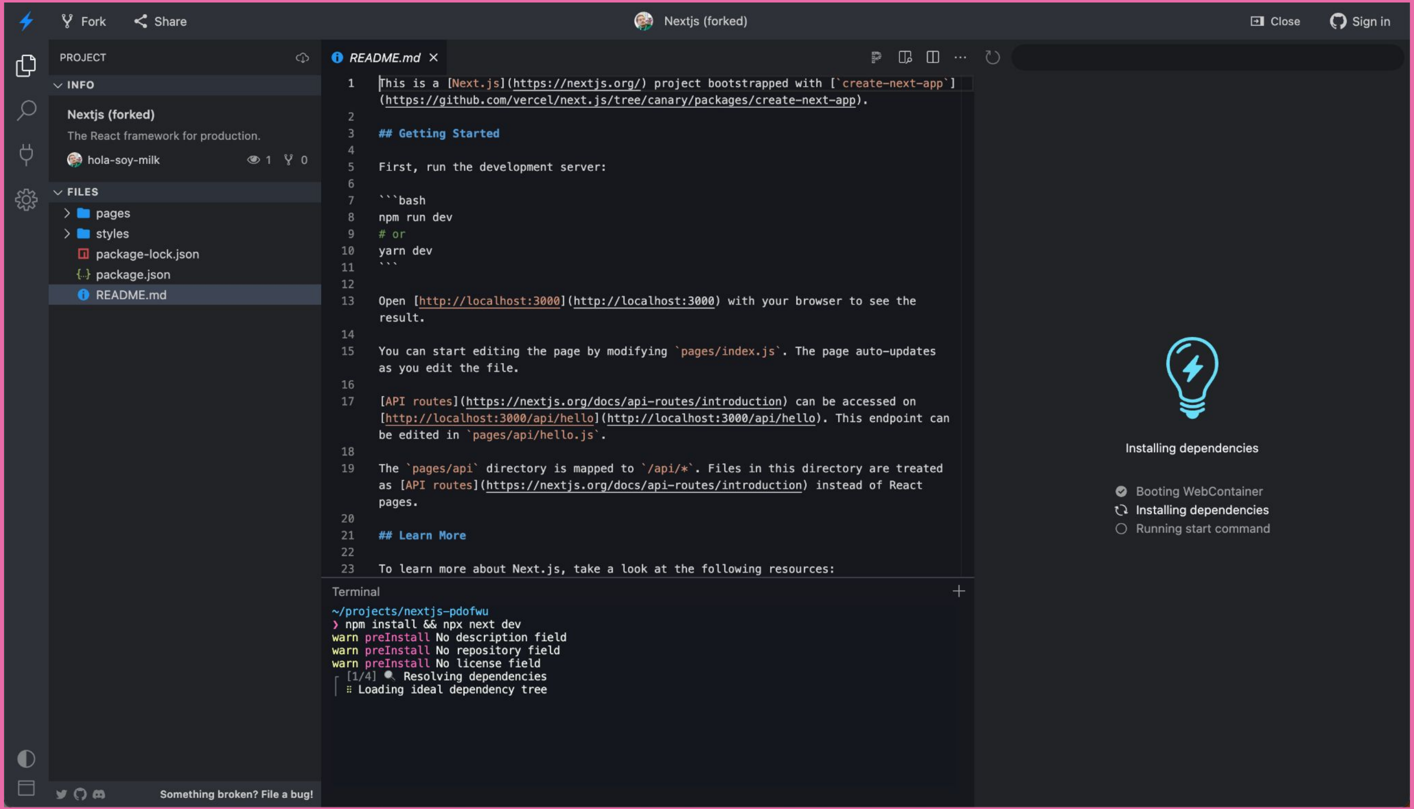
STREAM ONLY

https://archive.org/details/msdos_Prince_of_Persia_1990



Resize: off
Reduce palette: on
Colors: 239
Dithering: 0,18
Browser JPEG
Quality: 0,05
117 kB

https://squoosh.app/



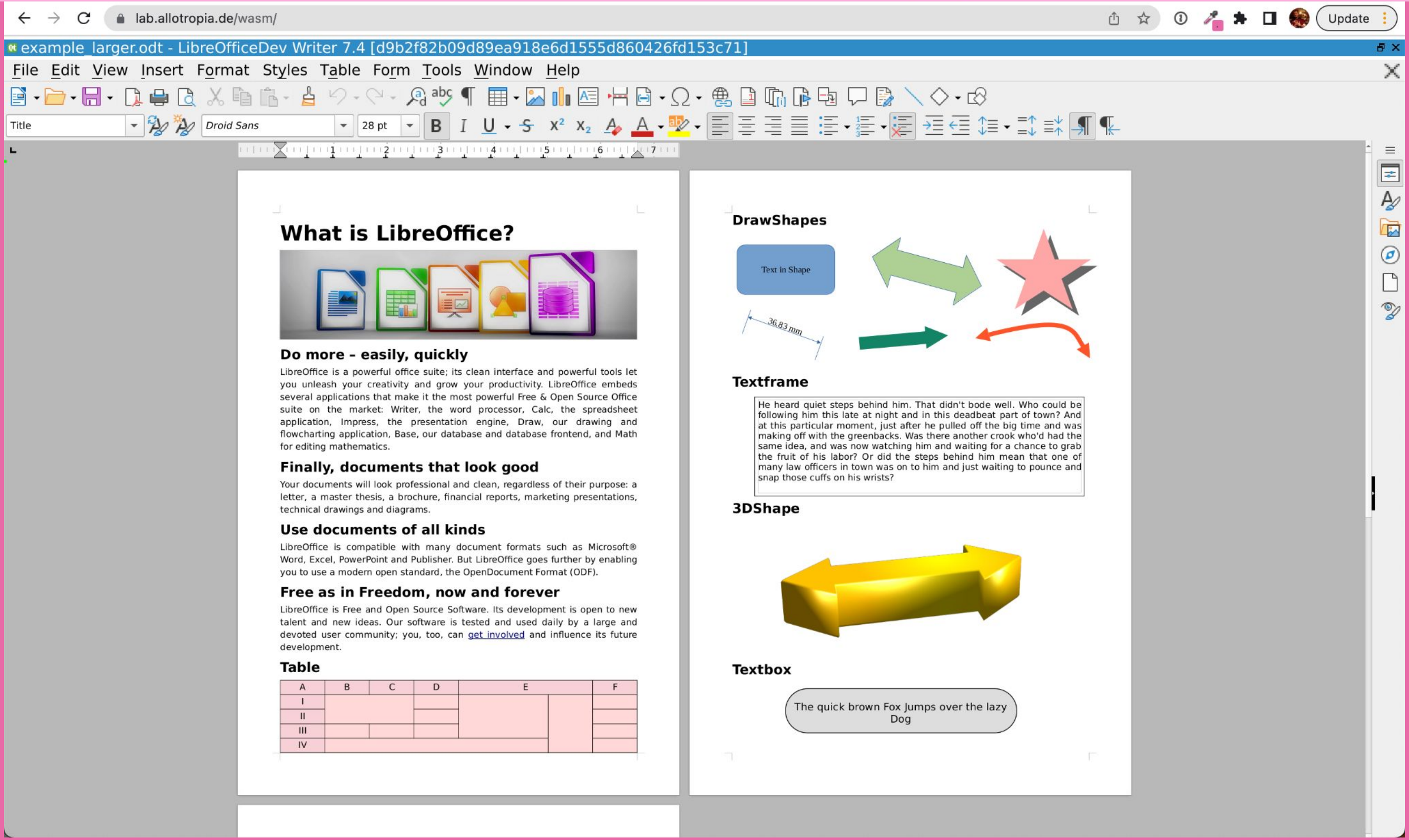
~~~
This is a [Next.js](https://nextjs.org/) project bootstrapped with [`create-next-app`](https://github.com/vercel/next.js/tree/canary/packages/create-next-app).

## Getting Started

First, run the development server:

```bash
npm run dev
# or
yarn dev
```

Open [http://localhost:3000](http://localhost:3000) with your browser to see the result.

You can start editing the page by modifying `pages/index.js`. The page auto-updates as you edit the file.

[API routes](https://nextjs.org/docs/api-routes/introduction) can be accessed on [http://localhost:3000/api/hello](http://localhost:3000/api/hello). This endpoint can be edited in `pages/api/hello.js`.

The `pages/api` directory is mapped to `/api/*`. Files in this directory are treated as [API routes](https://nextjs.org/docs/api-routes/introduction) instead of React pages.

## Learn More

To learn more about Next.js, take a look at the following resources:
~~~

Terminal

```
~/projects/nextjs-pdofwu
❯ npm install && npx next dev
warn preInstall No description field
warn preInstall No repository field
warn preInstall No license field
[1/4] 🔍 Resolving dependencies
⠿ Loading ideal dependency tree
```

https://stackblitz.com

Ramón Huidobro hola-soy-milk.online


lab.allotropia.de/wasm/
example_larger.odt - LibreOfficeDev Writer 7.4 [d9b2f82b09d89ea918e6d1555d860426fd153c71]
File Edit View Insert Format Styles Table Form Tools Window Help
Title
Droid Sans
28 pt
Update

What is LibreOffice?

Do more – easily, quickly

LibreOffice is a powerful office suite; its clean interface and powerful tools let you unleash your creativity and grow your productivity. LibreOffice embeds several applications that make it the most powerful Free & Open Source Office suite on the market: Writer, the word processor, Calc, the spreadsheet application, Impress, the presentation engine, Draw, our drawing and flowcharting application, Base, our database and database frontend, and Math for editing mathematics.

Finally, documents that look good

Your documents will look professional and clean, regardless of their purpose: a letter, a master thesis, a brochure, financial reports, marketing presentations, technical drawings and diagrams.

Use documents of all kinds

LibreOffice is compatible with many document formats such as Microsoft® Word, Excel, PowerPoint and Publisher. But LibreOffice goes further by enabling you to use a modern open standard, the OpenDocument Format (ODF).

Free as in Freedom, now and forever

LibreOffice is Free and Open Source Software. Its development is open to new talent and new ideas. Our software is tested and used daily by a large and devoted user community; you, too, can get involved and influence its future development.

Table

A	B	C	D	E		F
I						
II						
III						
IV						

DrawShapes

Textframe

He heard quiet steps behind him. That didn't bode well. Who could be following him this late at night and in this deadbeat part of town? And at this particular moment, just after he pulled off the big time and was making off with the greenbacks. Was there another crook who'd had the same idea, and was now watching him and waiting for a chance to grab the fruit of his labor? Or did the steps behind him mean that one of many law officers in town was on to him and just waiting to pounce and snap those cuffs on his wrists?

3DShape

Textbox

The quick brown Fox Jumps over the lazy Dog

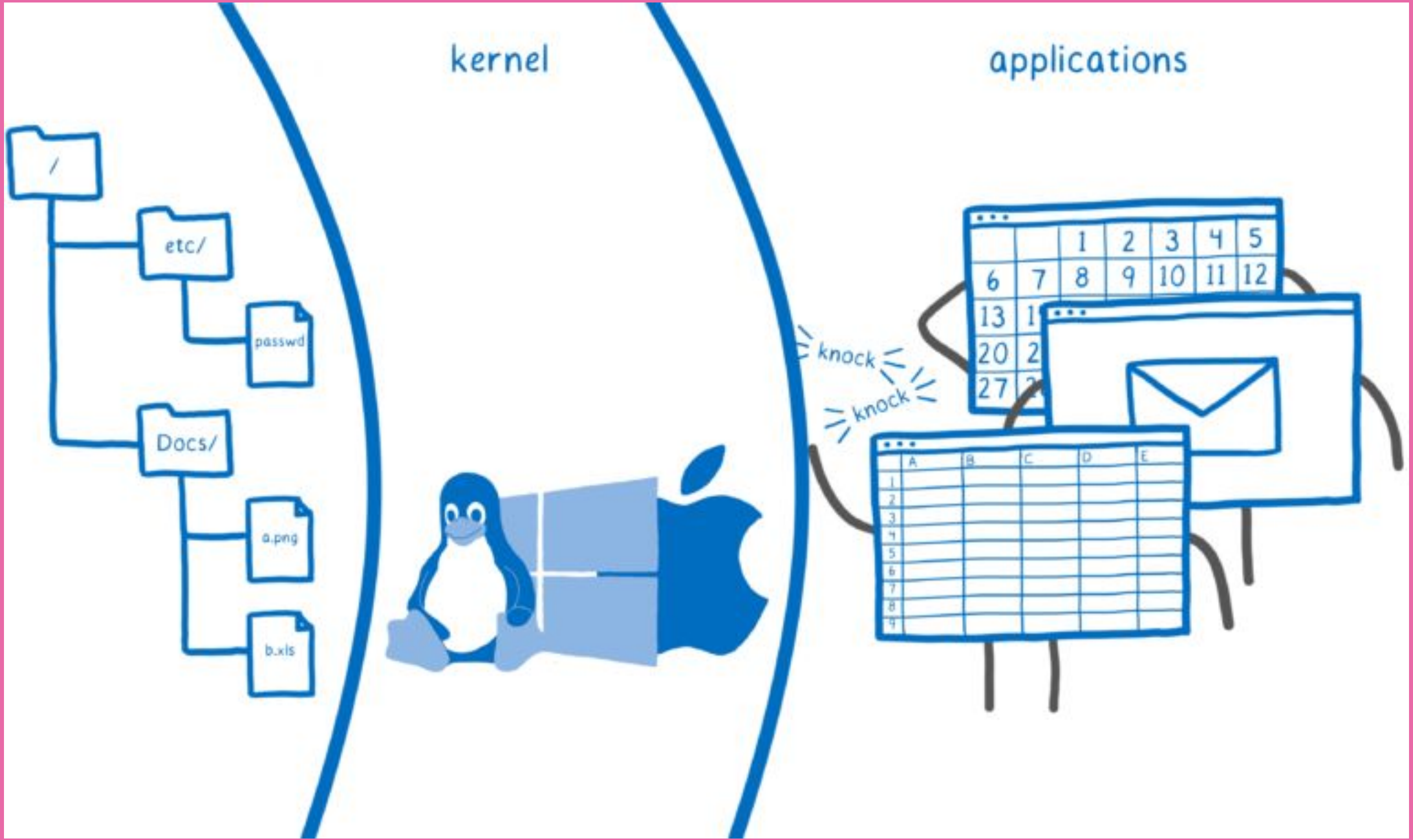
https://lab.allotropia.de/wasm/

Ok but that's all on the browser side. What's this about server-side Wasm?

Surely Wasm can't just run anywhere…

That's where WASI comes in.

Web**A**ssembly,
now with a
System **I**nterface!



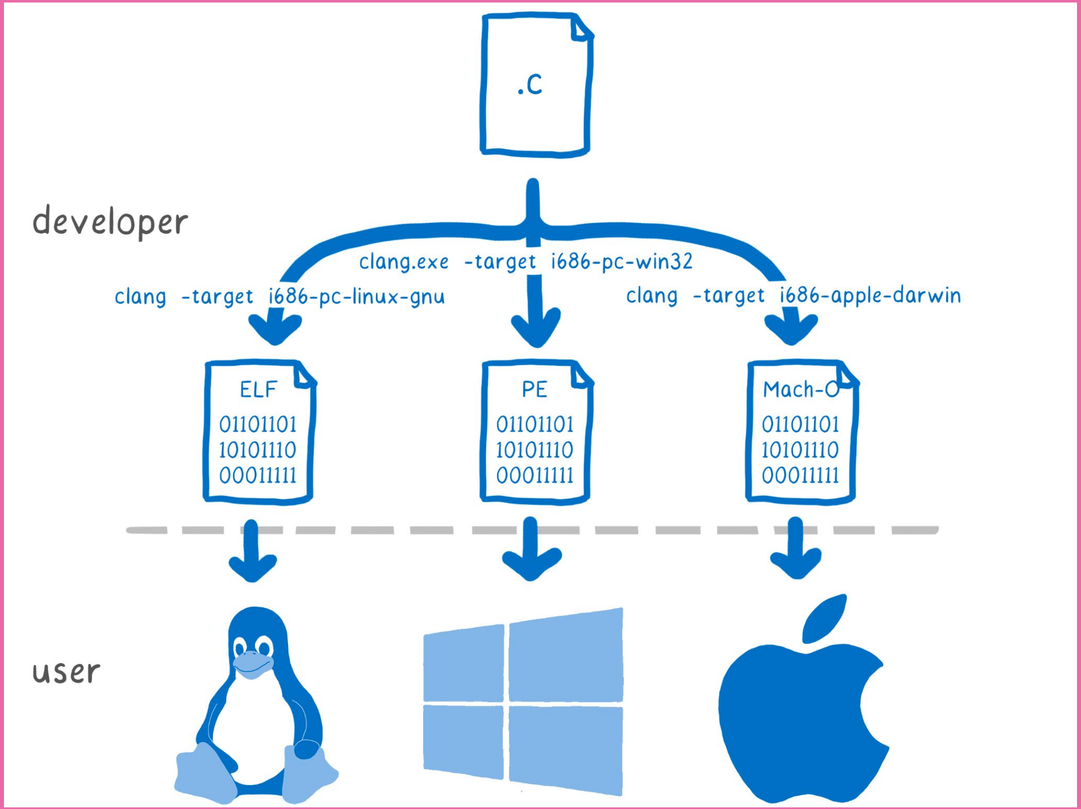
https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/

WASI is analogous to these system calls

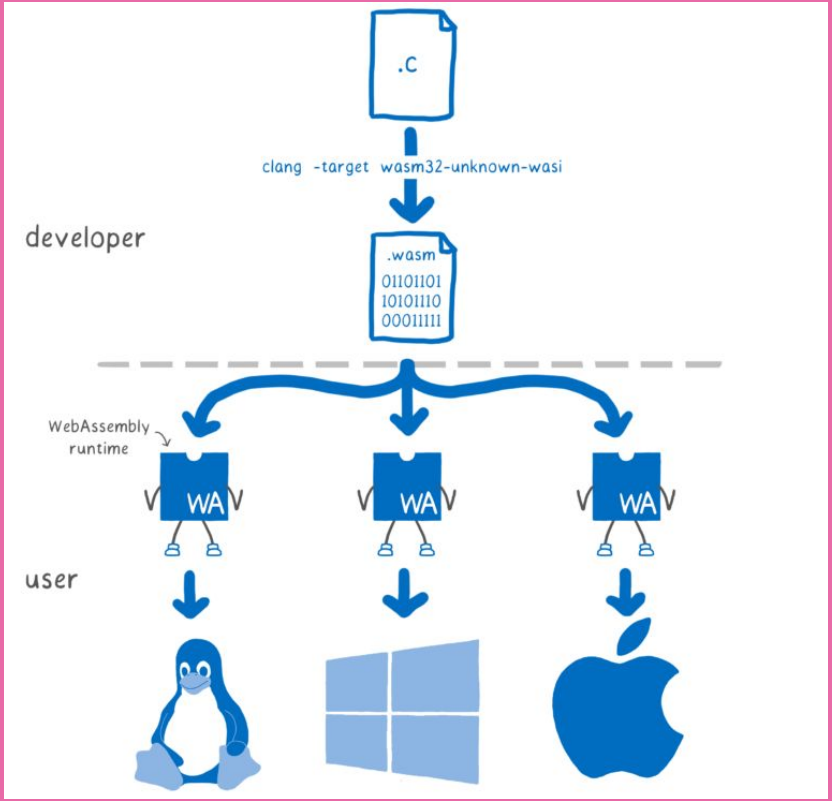
It is <u>not</u> an OS replacement

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...

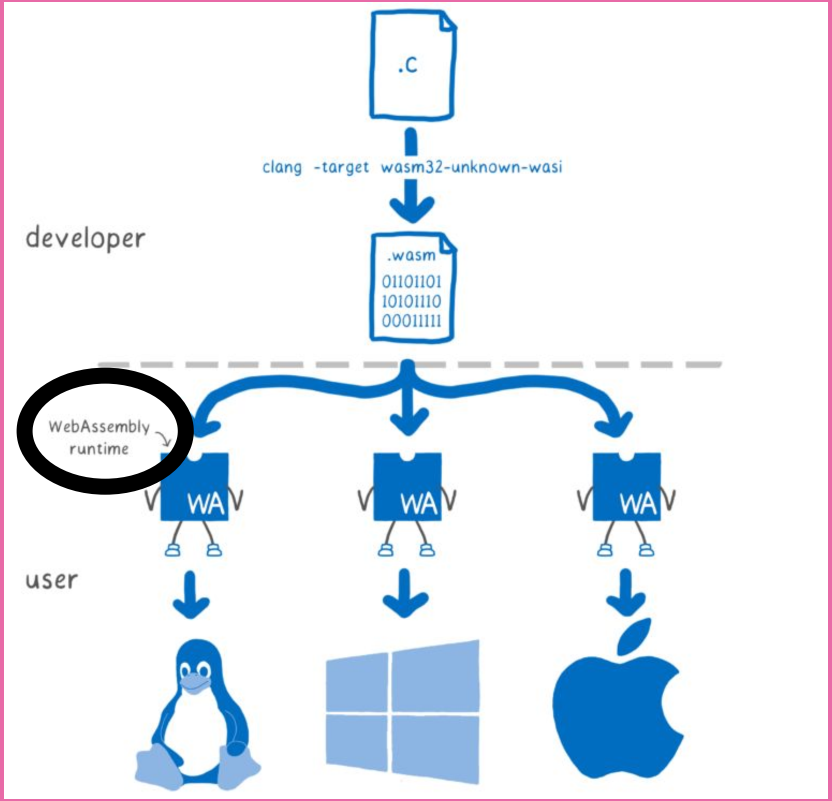
https://github.com/bytecodealliance/wasmtime/blob/main/docs/WASI-intro.md



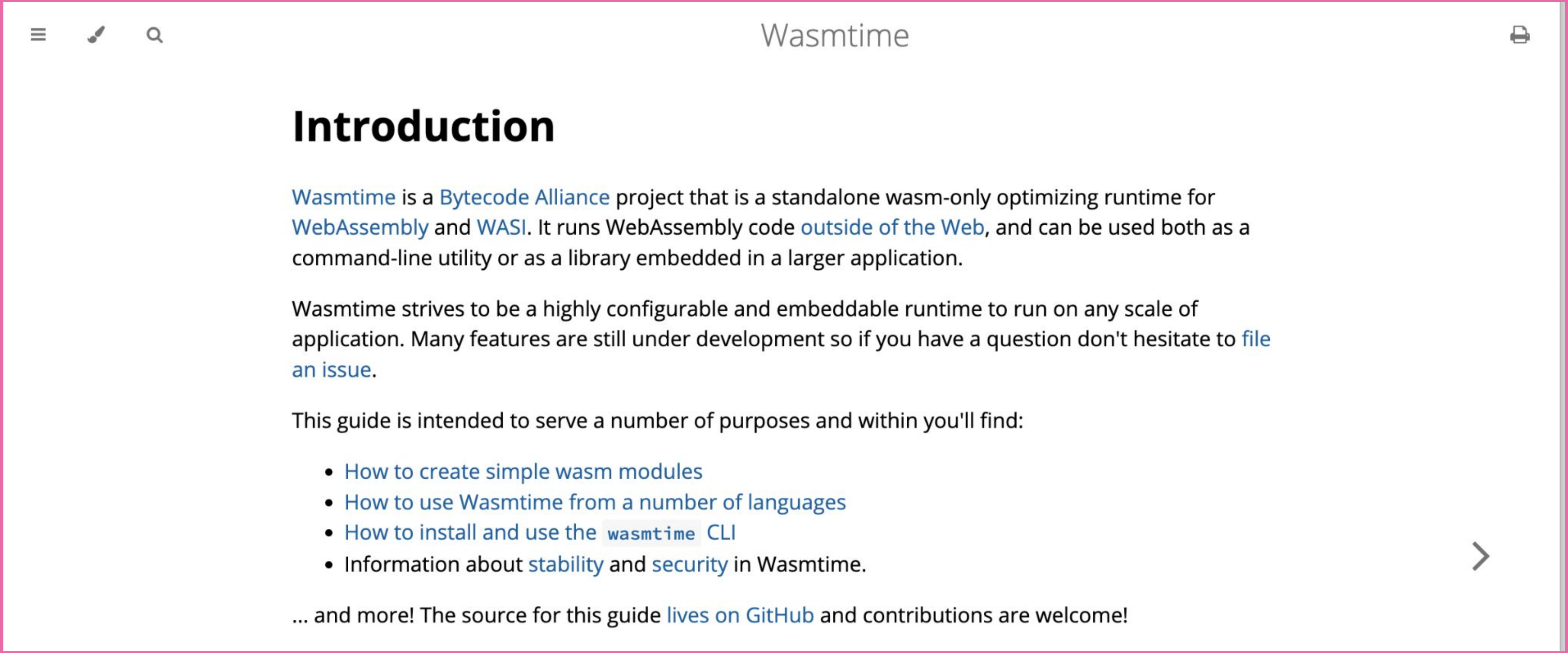
https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/



https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/



https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/



Wasmtime

Introduction

Wasmtime is a Bytecode Alliance project that is a standalone wasm-only optimizing runtime for WebAssembly and WASI. It runs WebAssembly code outside of the Web, and can be used both as a command-line utility or as a library embedded in a larger application.

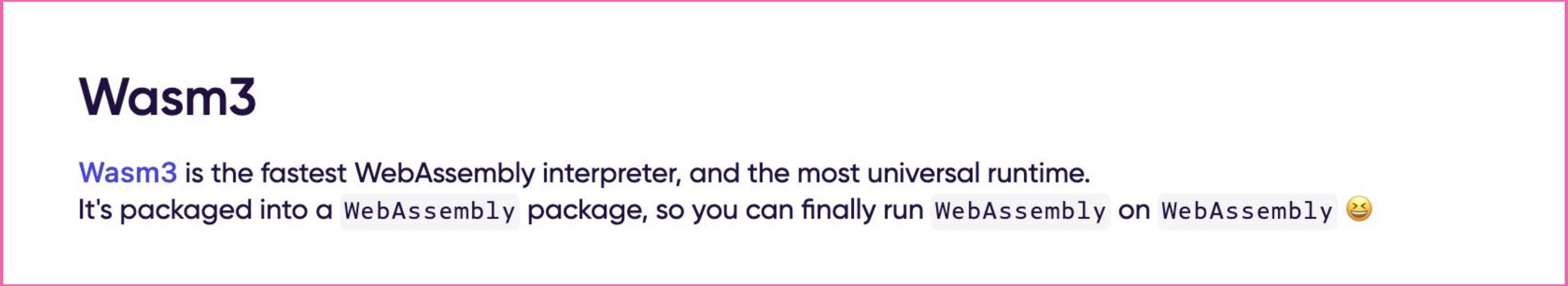
Wasmtime strives to be a highly configurable and embeddable runtime to run on any scale of application. Many features are still under development so if you have a question don't hesitate to file an issue.

This guide is intended to serve a number of purposes and within you'll find:

- How to create simple wasm modules
- How to use Wasmtime from a number of languages
- How to install and use the `wasmtime` CLI
- Information about stability and security in Wasmtime.

... and more! The source for this guide lives on GitHub and contributions are welcome!

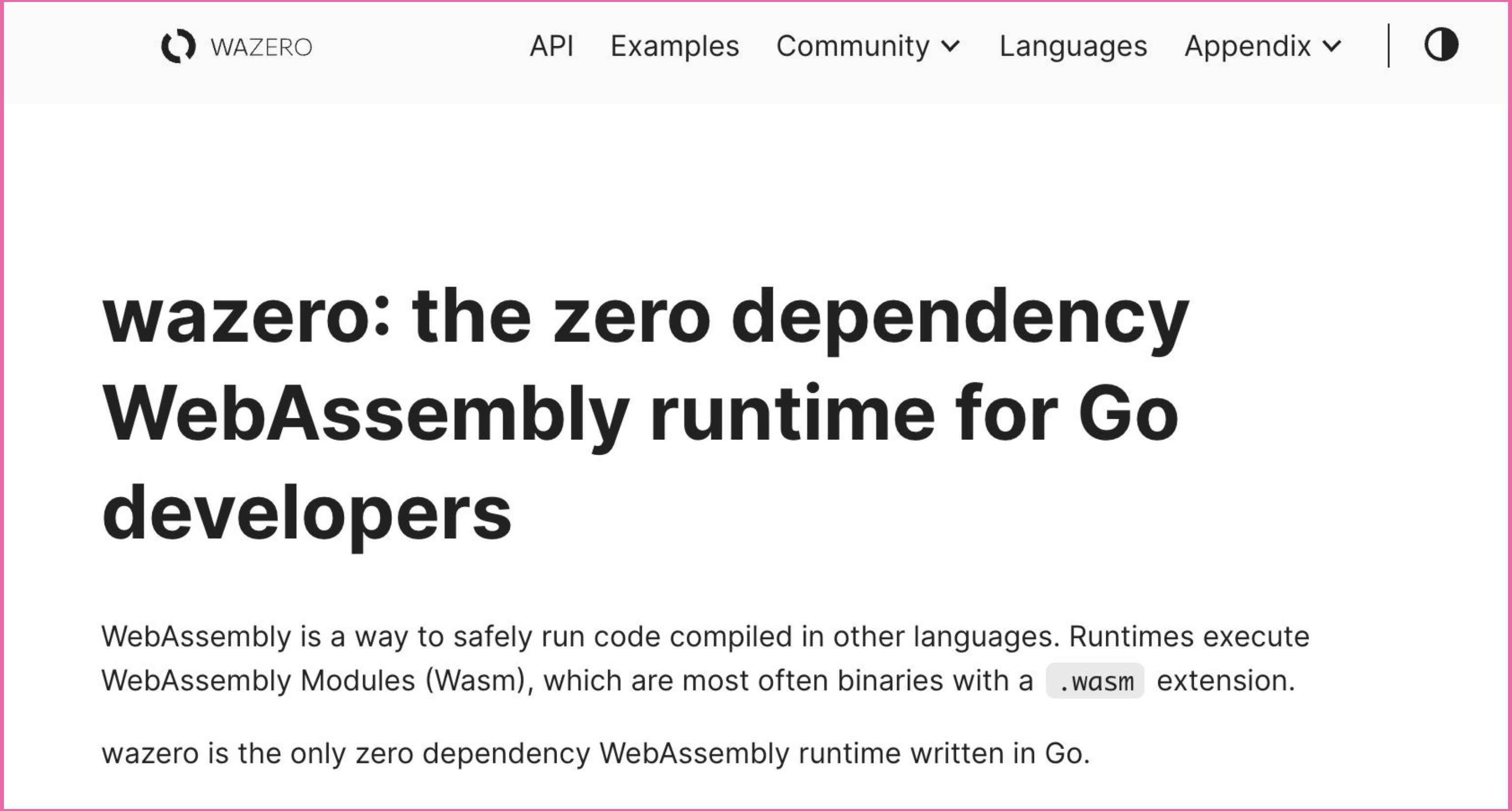
https://docs.wasmtime.dev/



Wasm3

Wasm3 is the fastest WebAssembly interpreter, and the most universal runtime.
It's packaged into a `WebAssembly` package, so you can finally run `WebAssembly` on `WebAssembly` 😆

https://wapm.io/vshymanskyy/wasm3

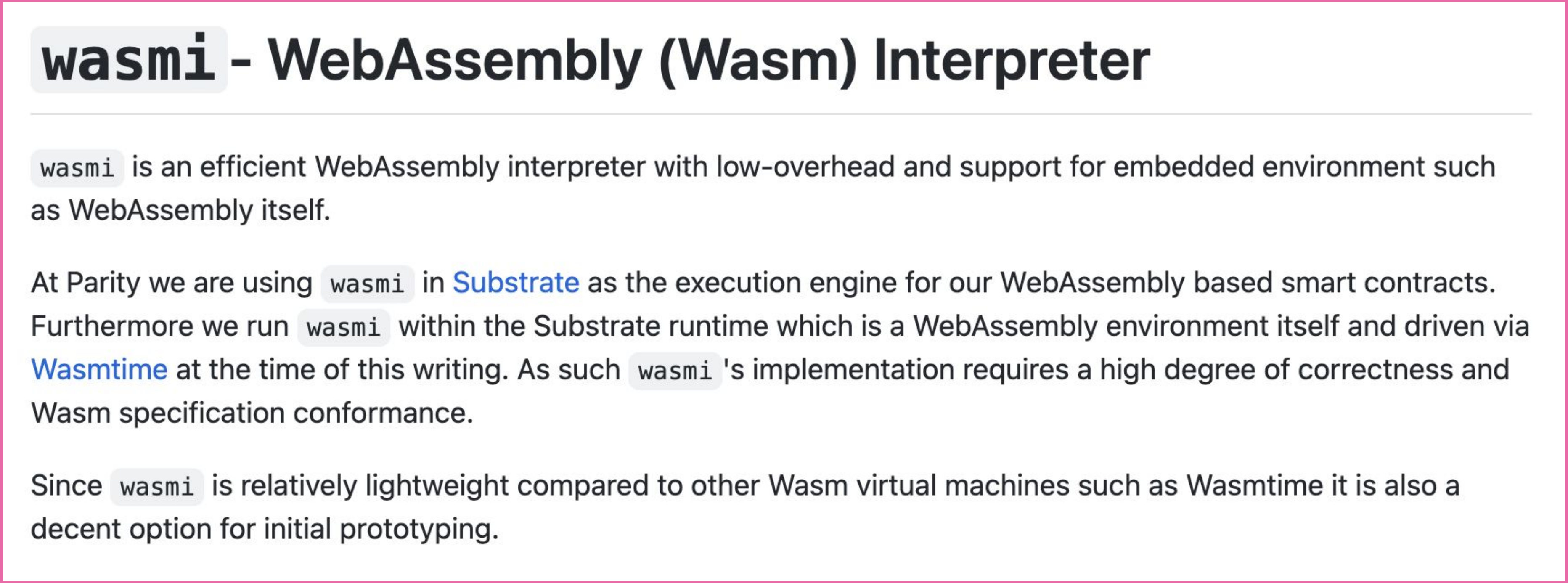


wazero: the zero dependency WebAssembly runtime for Go developers

WebAssembly is a way to safely run code compiled in other languages. Runtimes execute WebAssembly Modules (Wasm), which are most often binaries with a `.wasm` extension.

wazero is the only zero dependency WebAssembly runtime written in Go.

https://wazero.io/



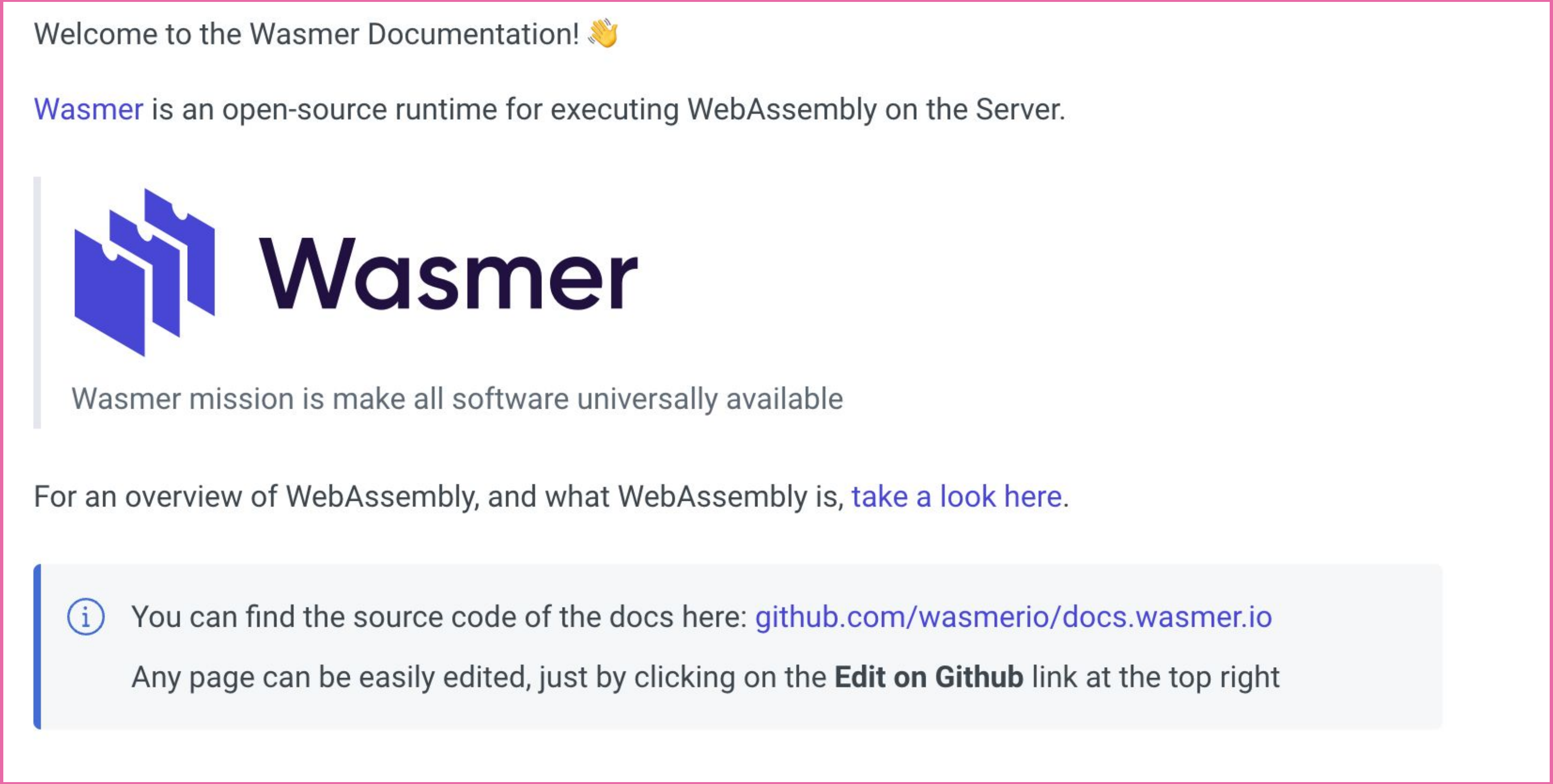
wasmi - WebAssembly (Wasm) Interpreter

wasmi is an efficient WebAssembly interpreter with low-overhead and support for embedded environment such as WebAssembly itself.

At Parity we are using wasmi in Substrate as the execution engine for our WebAssembly based smart contracts. Furthermore we run wasmi within the Substrate runtime which is a WebAssembly environment itself and driven via Wasmtime at the time of this writing. As such wasmi's implementation requires a high degree of correctness and Wasm specification conformance.

Since wasmi is relatively lightweight compared to other Wasm virtual machines such as Wasmtime it is also a decent option for initial prototyping.

https://github.com/paritytech/wasmi



Welcome to the Wasmer Documentation! 👋

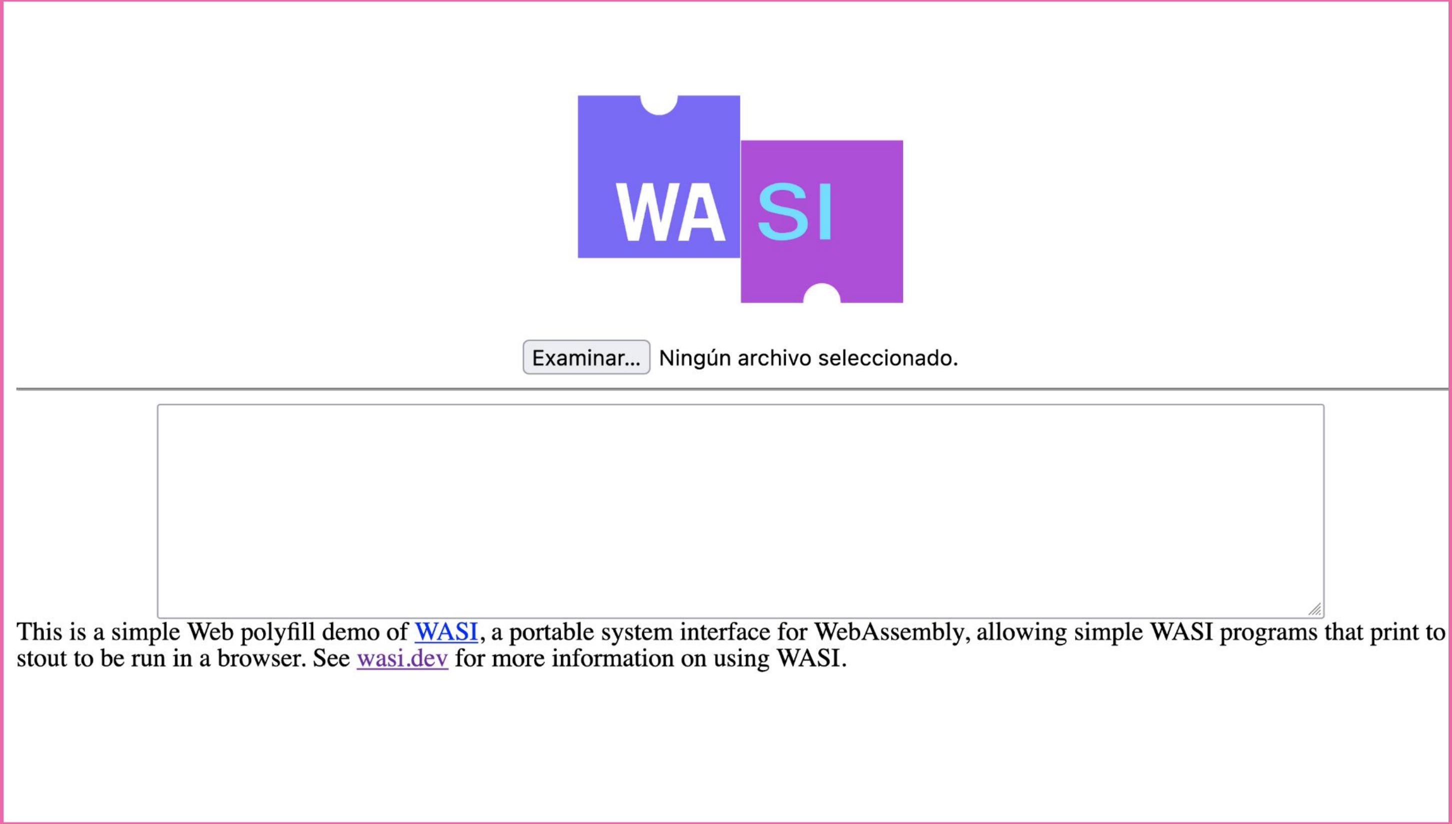
Wasmer is an open-source runtime for executing WebAssembly on the Server.

> Wasmer
>
> Wasmer mission is make all software universally available

For an overview of WebAssembly, and what WebAssembly is, take a look here.

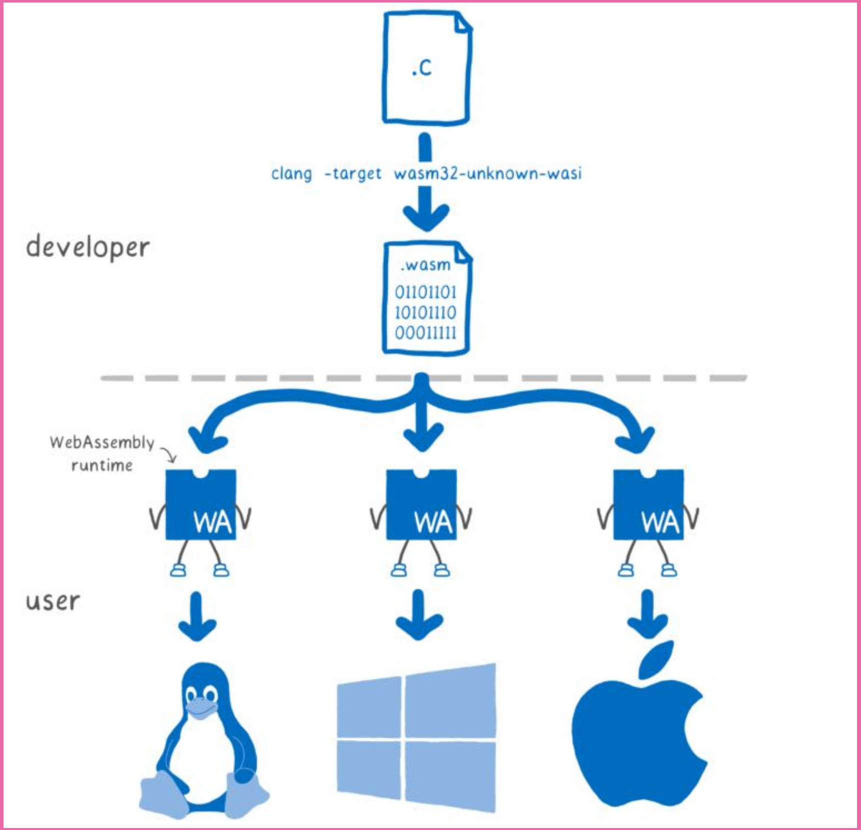
> ⓘ You can find the source code of the docs here: github.com/wasmerio/docs.wasmer.io
>
> Any page can be easily edited, just by clicking on the **Edit on Github** link at the top right

https://wasmer.io/

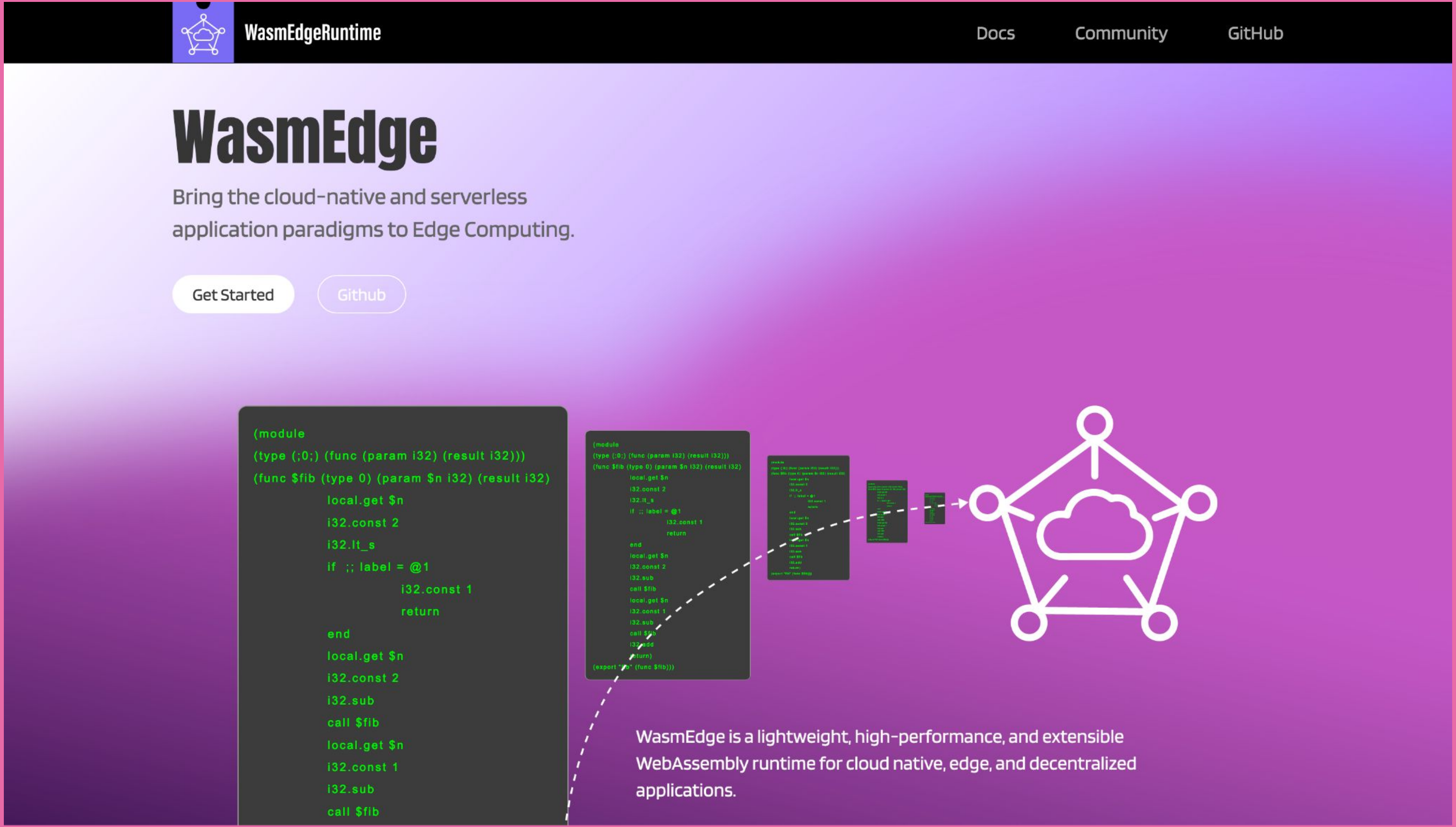


This is a simple Web polyfill demo of WASI, a portable system interface for WebAssembly, allowing simple WASI programs that print to stout to be run in a browser. See wasi.dev for more information on using WASI.

https://wasi.dev/polyfill/



https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/



WasmEdge

Bring the cloud-native and serverless application paradigms to Edge Computing.

```
(module
(type (;0;) (func (param i32) (result i32)))
(func $fib (type 0) (param $n i32) (result i32)
          local.get $n
          i32.const 2
          i32.lt_s
          if  ;; label = @1
                    i32.const 1
                    return
          end
          local.get $n
          i32.const 2
          i32.sub
          call $fib
          local.get $n
          i32.const 1
          i32.sub
          call $fib
```

WasmEdge is a lightweight, high-performance, and extensible WebAssembly runtime for cloud native, edge, and decentralized applications.

https://wasmedge.org/

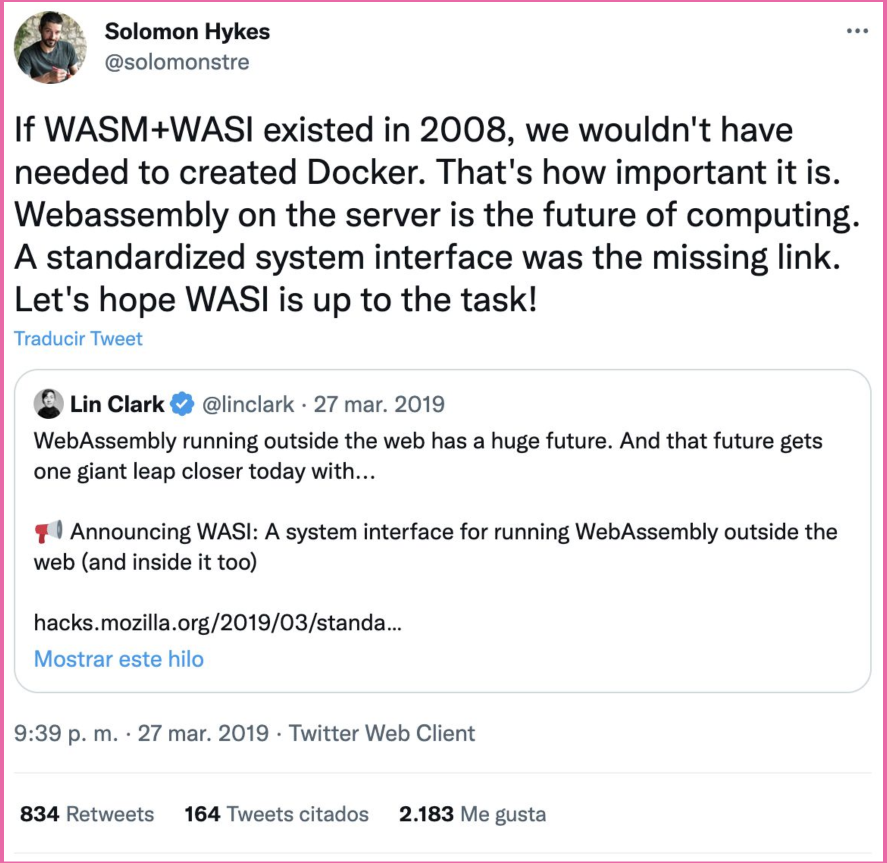
Or even… in a container 🤯

Further Reading/Watching

- https://developer.mozilla.org/en-US/docs/WebAssembly
- https://hacks.mozilla.org/2017/02/a-cartoon-intro-to-webassembly/
- https://github.com/bytecodealliance/wasmtime/blob/main/docs/WASI-overview.md
- https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/

Why run Wasm on the server?

WA



Solomon Hykes
@solomonstre

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

Traducir Tweet

Lin Clark @linclark · 27 mar. 2019

WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with…

📢 Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)

hacks.mozilla.org/2019/03/standa…

Mostrar este hilo

9:39 p. m. · 27 mar. 2019 · Twitter Web Client

834 Retweets **164** Tweets citados **2.183** Me gusta

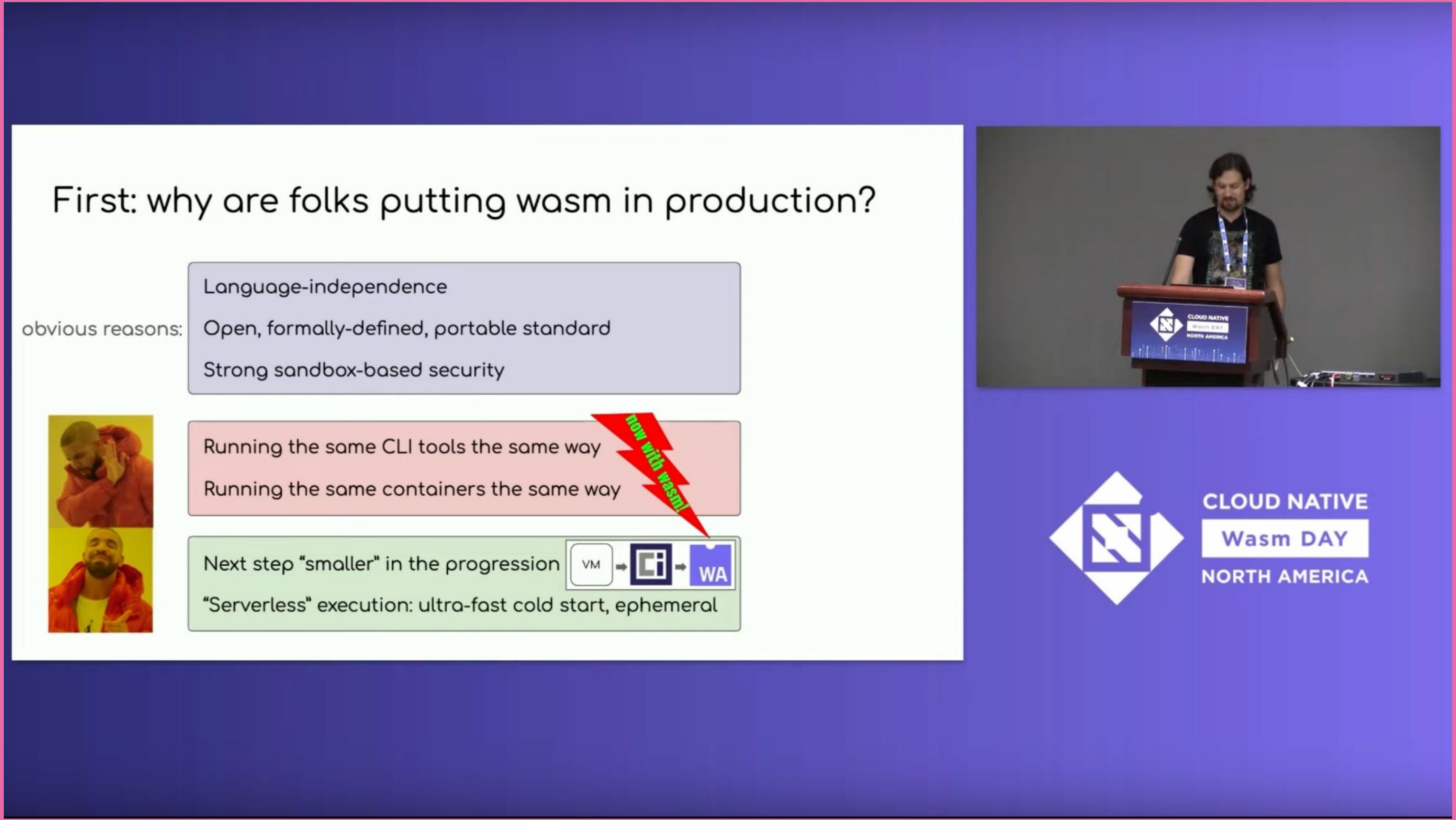
https://twitter.com/solomonstre/status/1111004913222324225

It’s designed with capability-based security!

It’s polyglot by nature!

Modules are typed, small, provisionable!

It’s got the speed!



First: why are folks putting wasm in production?

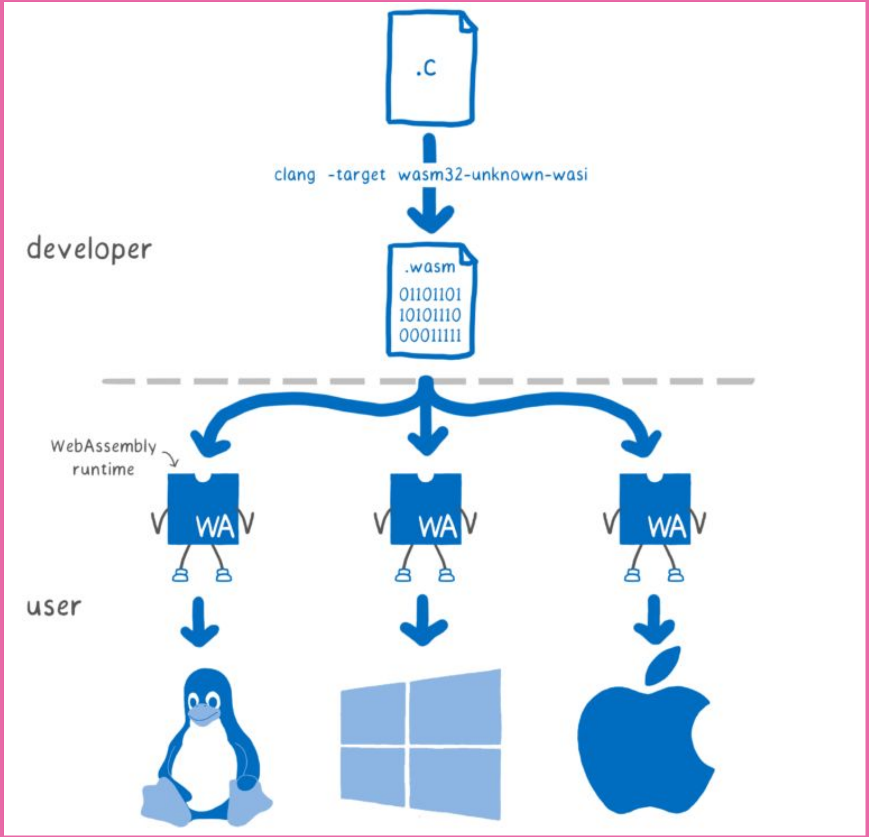
obvious reasons:

- Language-independence
- Open, formally-defined, portable standard
- Strong sandbox-based security

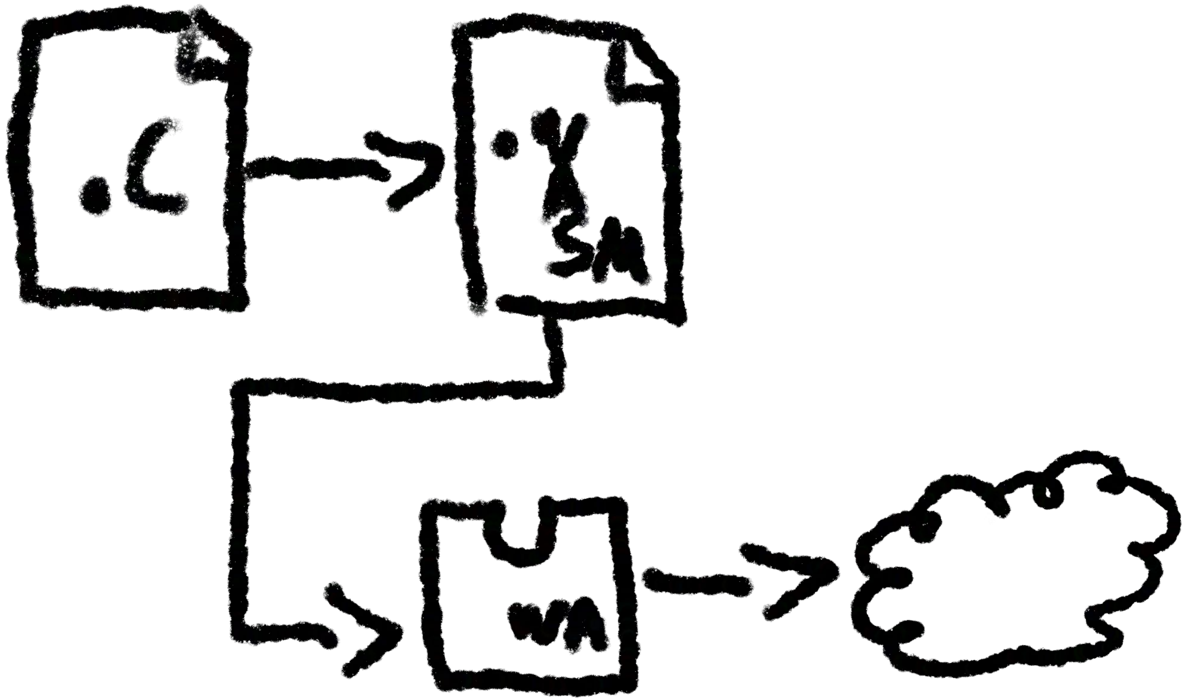
- Running the same CLI tools the same way
- Running the same containers the same way

- Next step "smaller" in the progression
- "Serverless" execution: ultra-fast cold start, ephemeral

https://www.youtube.com/watch?v=phodPLY8zNE



https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/


.C
.~~X~~ ASM
WA

Code attaches to system resources at startup.

Code attaches to system resources at startup.

But what does that mean for the industry?

We can write code without worrying about

Server Setup

We can write code without worrying about

Startup/Shutdown times

We can write code without worrying about

Scaling

We can write code without worrying about

Common Security Vectors

Further Reading/Watching

- https://www.secondstate.io/articles/why-webassembly-server/
- https://wasmedge.org/book/en/use_cases/server_side_render.html
- https://www.wasm.builders/thomastaylor312/why-webassembly-belongs-outside-the-browser-331a

Is Wasm production-ready?

Functions-as-a-Service (Faas)

fastly

SECOND STATE

CLOUDFLARE

Edge Computing / Microservices

wasmcloud

FERMYON

CLOUD NATIVE COMPUTING FOUNDATION

Vercel

Extensibility

Figma

Istio

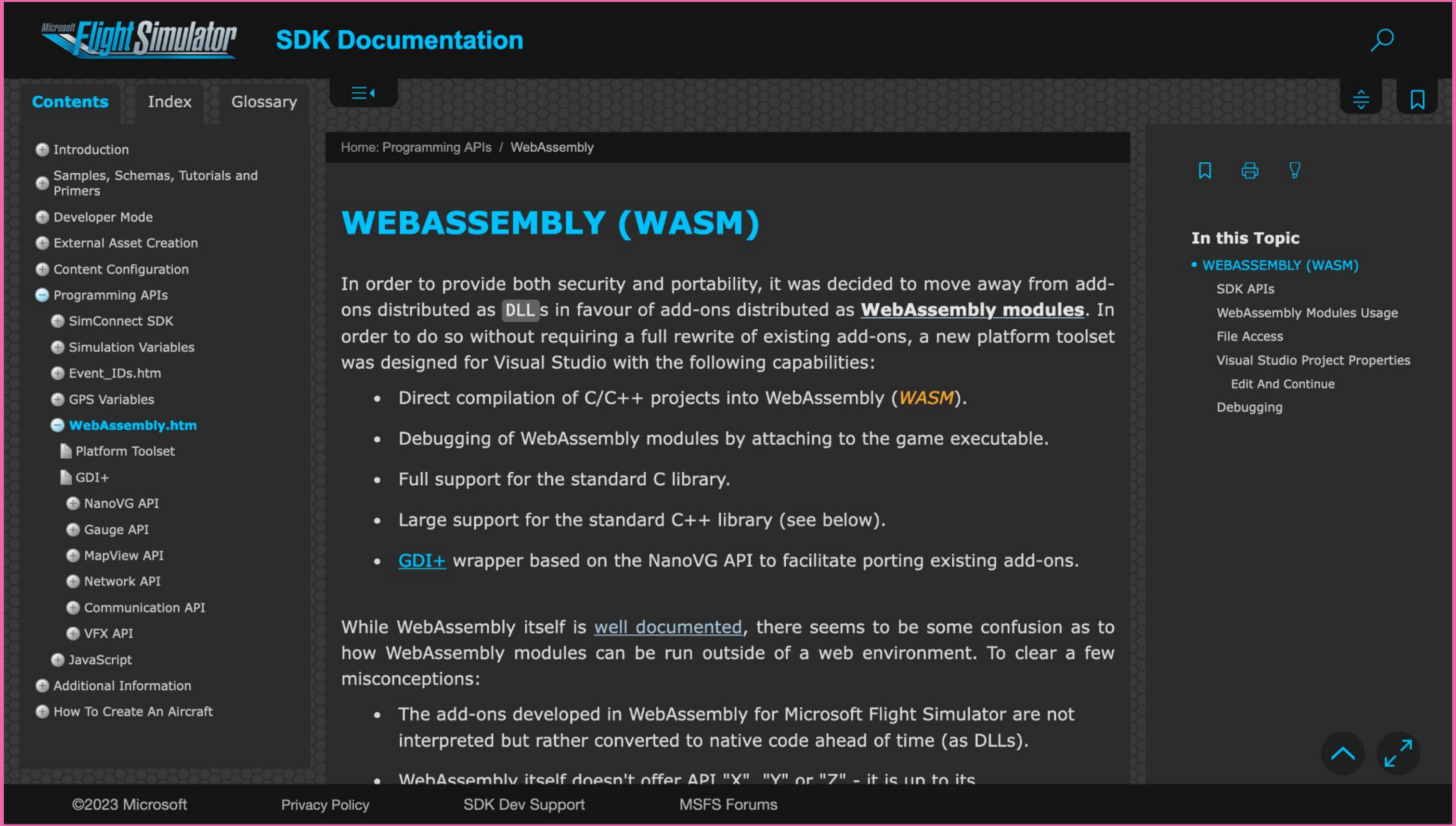
shopify

Blockchain

parity

Embedded

Akri



Home: Programming APIs / WebAssembly

WEBASSEMBLY (WASM)

In order to provide both security and portability, it was decided to move away from add-ons distributed as `DLL`s in favour of add-ons distributed as **WebAssembly modules**. In order to do so without requiring a full rewrite of existing add-ons, a new platform toolset was designed for Visual Studio with the following capabilities:

- Direct compilation of C/C++ projects into WebAssembly (*WASM*).
- Debugging of WebAssembly modules by attaching to the game executable.
- Full support for the standard C library.
- Large support for the standard C++ library (see below).
- GDI+ wrapper based on the NanoVG API to facilitate porting existing add-ons.

While WebAssembly itself is well documented, there seems to be some confusion as to how WebAssembly modules can be run outside of a web environment. To clear a few misconceptions:

- The add-ons developed in WebAssembly for Microsoft Flight Simulator are not interpreted but rather converted to native code ahead of time (as DLLs).
- WebAssembly itself doesn't offer API "X", "Y" or "Z" - it is up to its

In this Topic

https://docs.flightsimulator.com/html/Programming_Tools/WASM/WebAssembly.htm

Further Reading/Watching

- https://shopify.engineering/shopify-webassembly
- https://blog.suborbital.dev/webassembly-extensibility-today-and-tomorrow
- https://www.wasm.builders/aryank21/why-wasm-is-the-perfect-runtime-for-server-side-applications-1b9p

But wait there’s more:

- Component model

But wait there's more:

- Component model
- wasi-nn

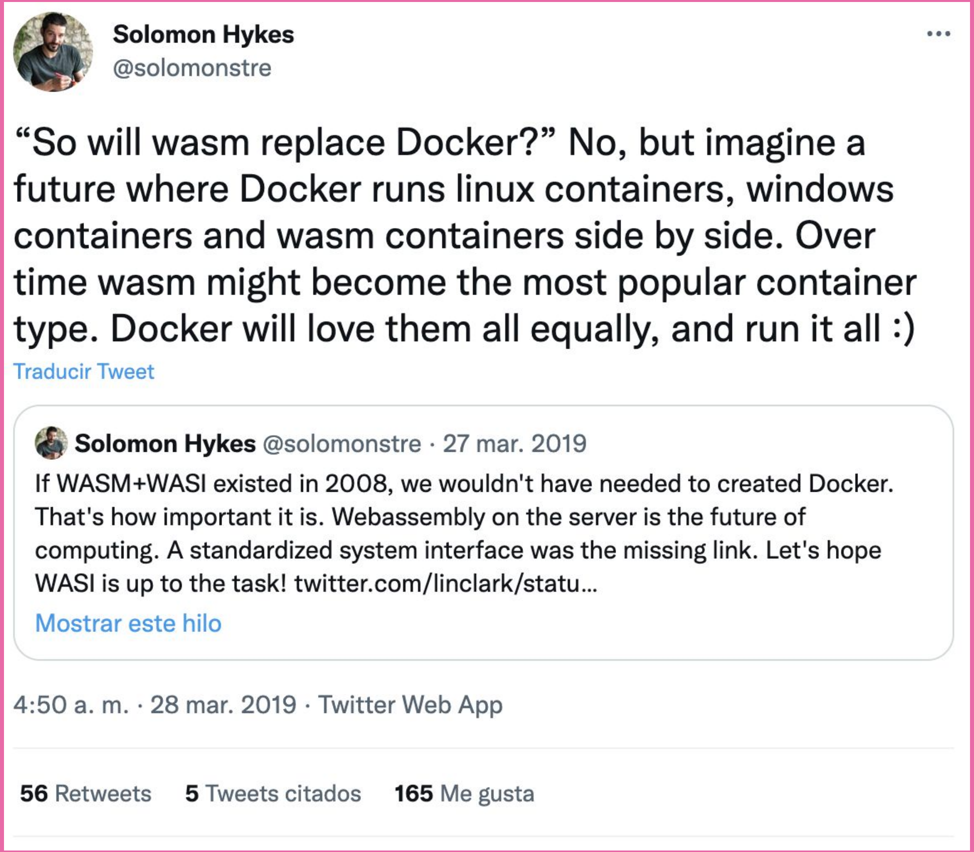
But wait there's more:

- Component model
- wasi-nn
- Garbage Collection

But wait there's more:

- Component model
- wasi-nn
- Garbage Collection
- Multi-threading

Who wants Docker+Wasm? 🙋



Solomon Hykes
@solomonstre

“So will wasm replace Docker?” No, but imagine a future where Docker runs linux containers, windows containers and wasm containers side by side. Over time wasm might become the most popular container type. Docker will love them all equally, and run it all :)

Traducir Tweet

Solomon Hykes @solomonstre · 27 mar. 2019
If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task! twitter.com/linclark/statu...

Mostrar este hilo

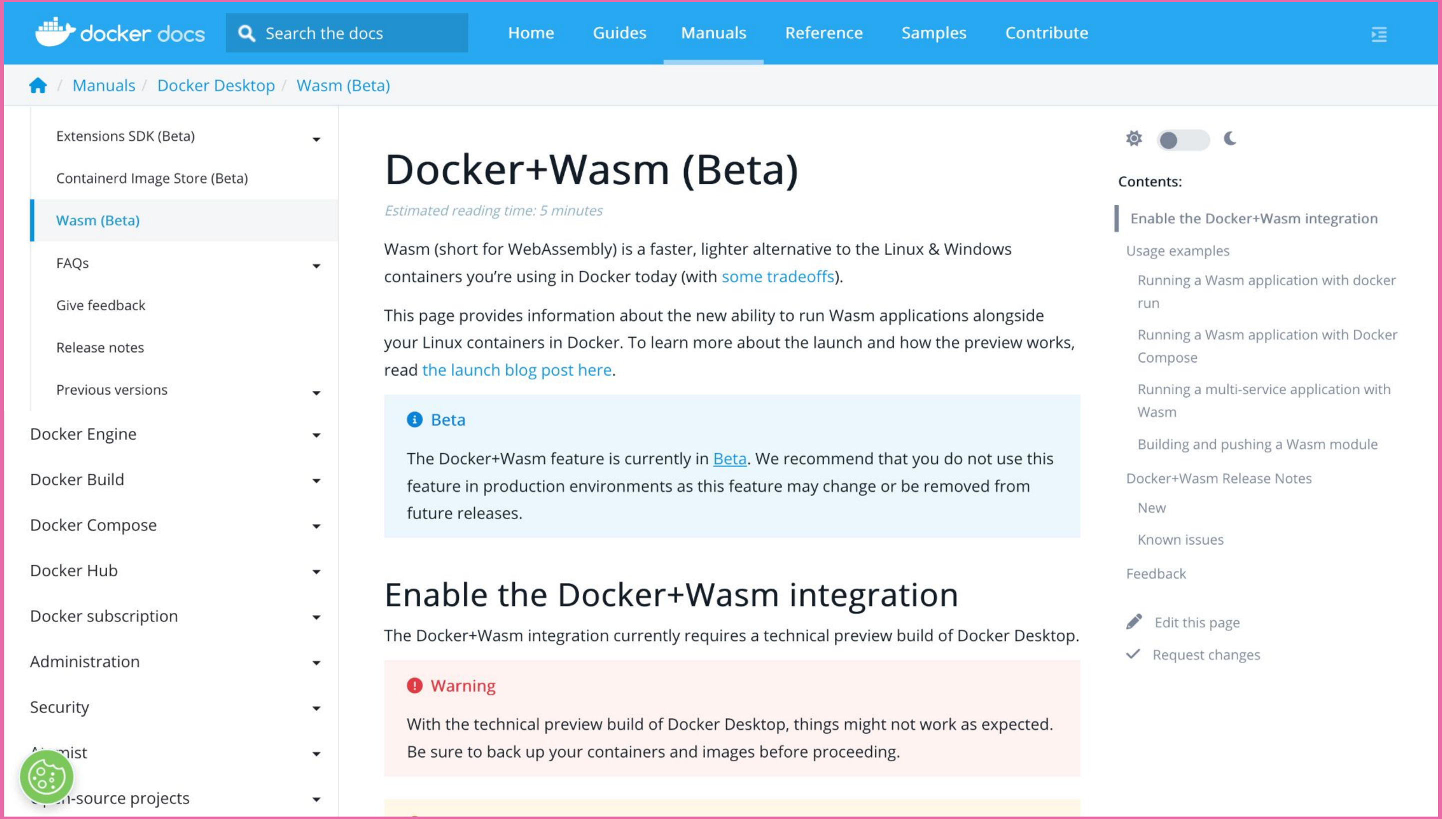
4:50 a. m. · 28 mar. 2019 · Twitter Web App

56 Retweets 5 Tweets citados 165 Me gusta

https://twitter.com/solomonstre/status/1111113329647325185

Ramón Huidobro

hola-soy-milk.online



Docker+Wasm (Beta)

Estimated reading time: 5 minutes

Wasm (short for WebAssembly) is a faster, lighter alternative to the Linux & Windows containers you're using in Docker today (with some tradeoffs).

This page provides information about the new ability to run Wasm applications alongside your Linux containers in Docker. To learn more about the launch and how the preview works, read the launch blog post here.

Beta

The Docker+Wasm feature is currently in Beta. We recommend that you do not use this feature in production environments as this feature may change or be removed from future releases.

Enable the Docker+Wasm integration

The Docker+Wasm integration currently requires a technical preview build of Docker Desktop.

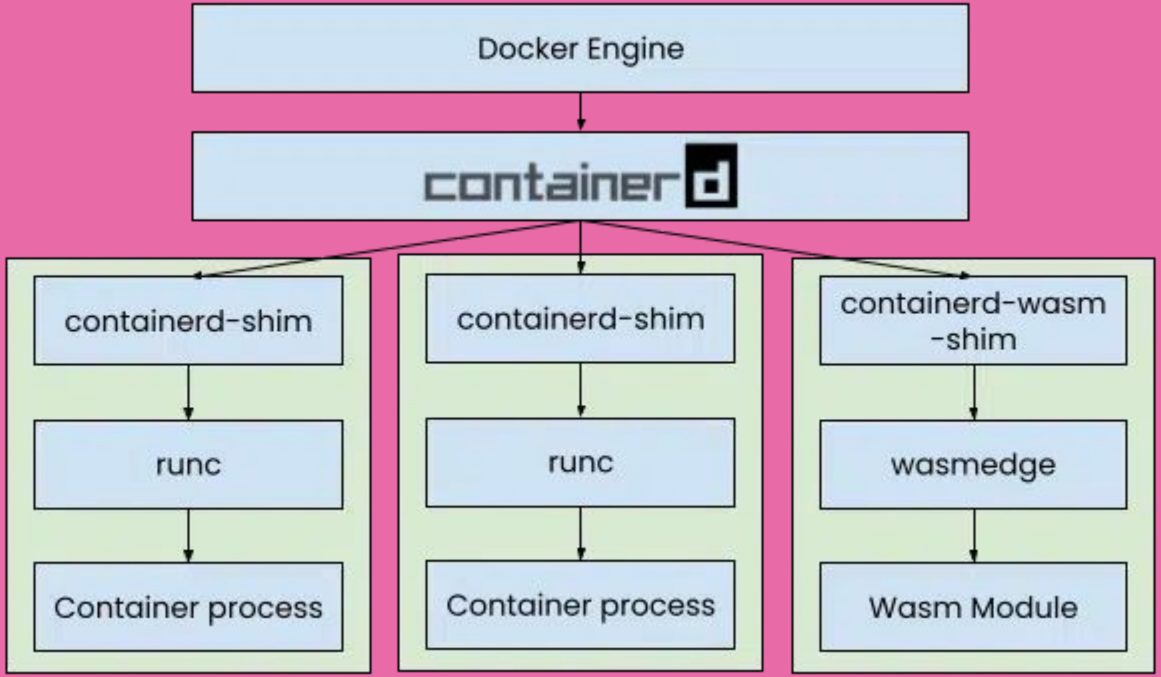
Warning

With the technical preview build of Docker Desktop, things might not work as expected. Be sure to back up your containers and images before proceeding.

Contents:

Edit this page
Request changes

https://docs.docker.com/desktop/wasm/



https://www.docker.com/blog/docker-wasm-technical-preview/

A lot of this is new and WIP...

But the future is very bright!

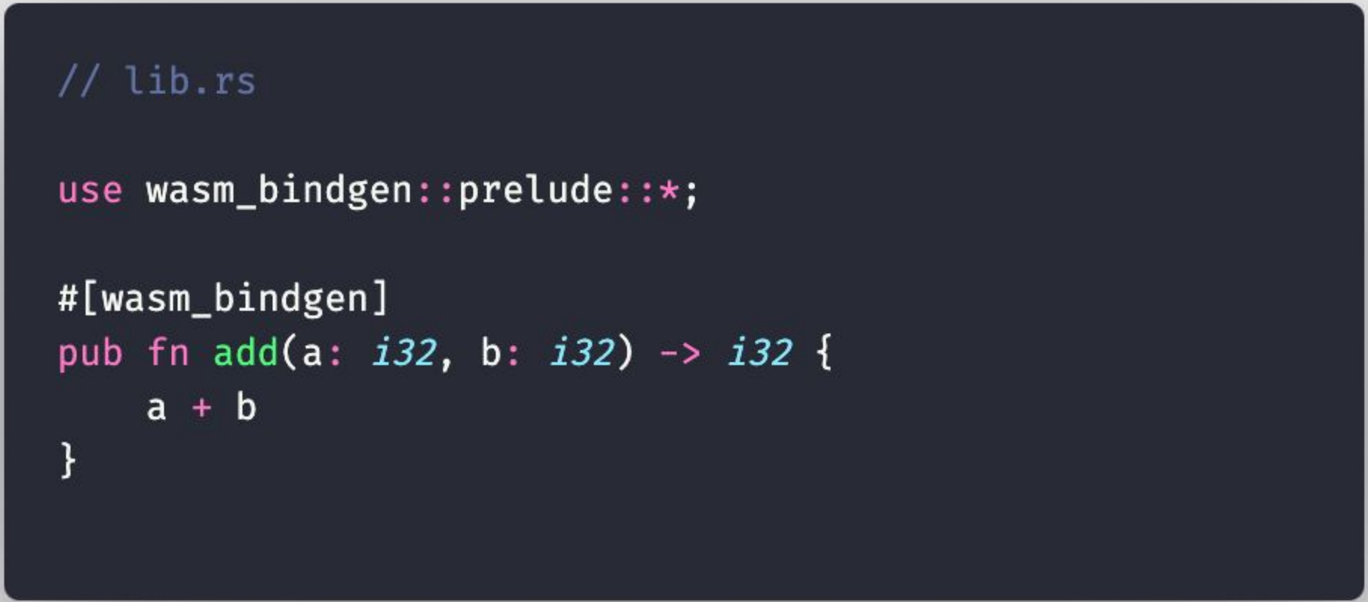
Hold on.

Hold on.

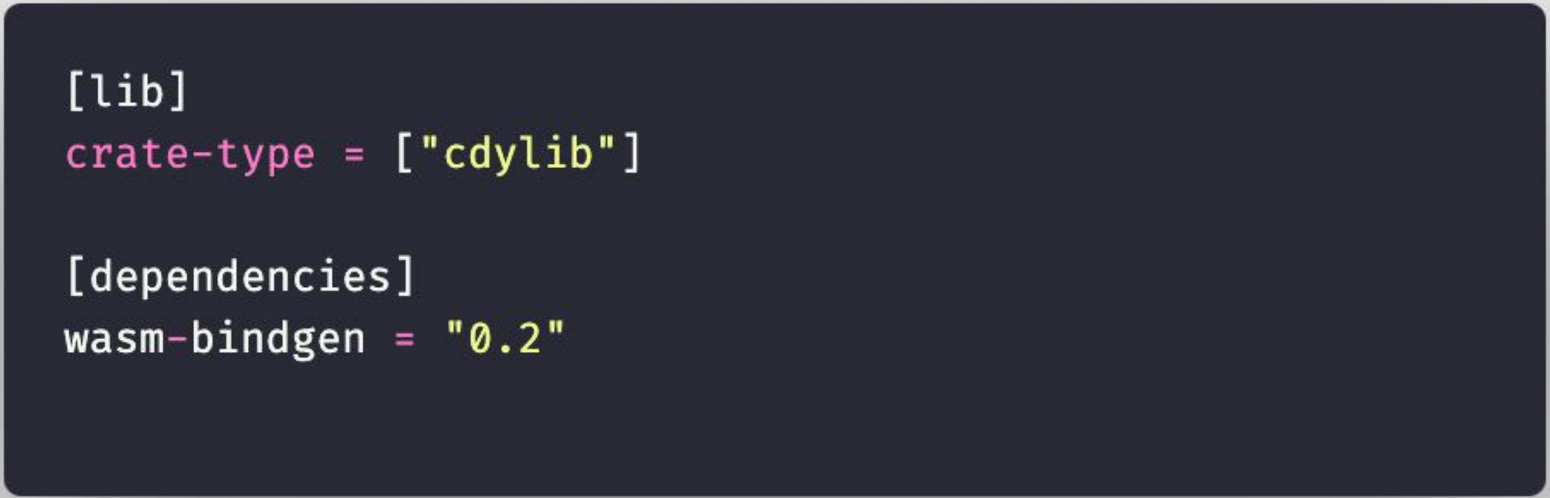
Where does React come in?

Hold on.

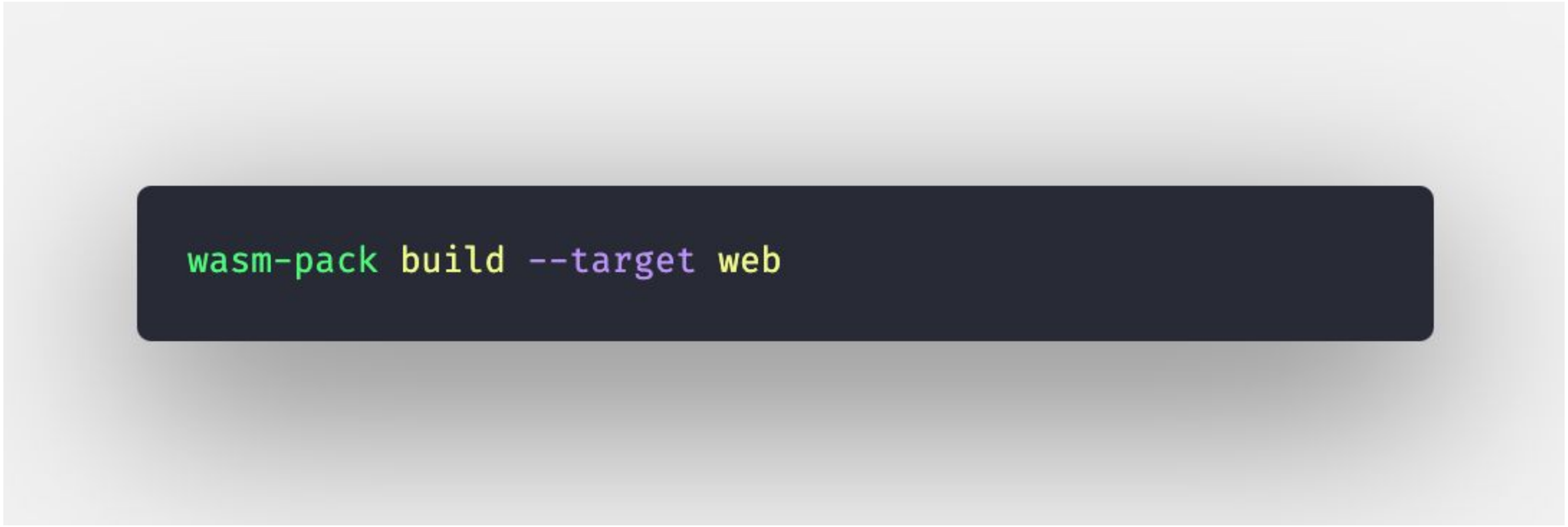
Where does React come in?



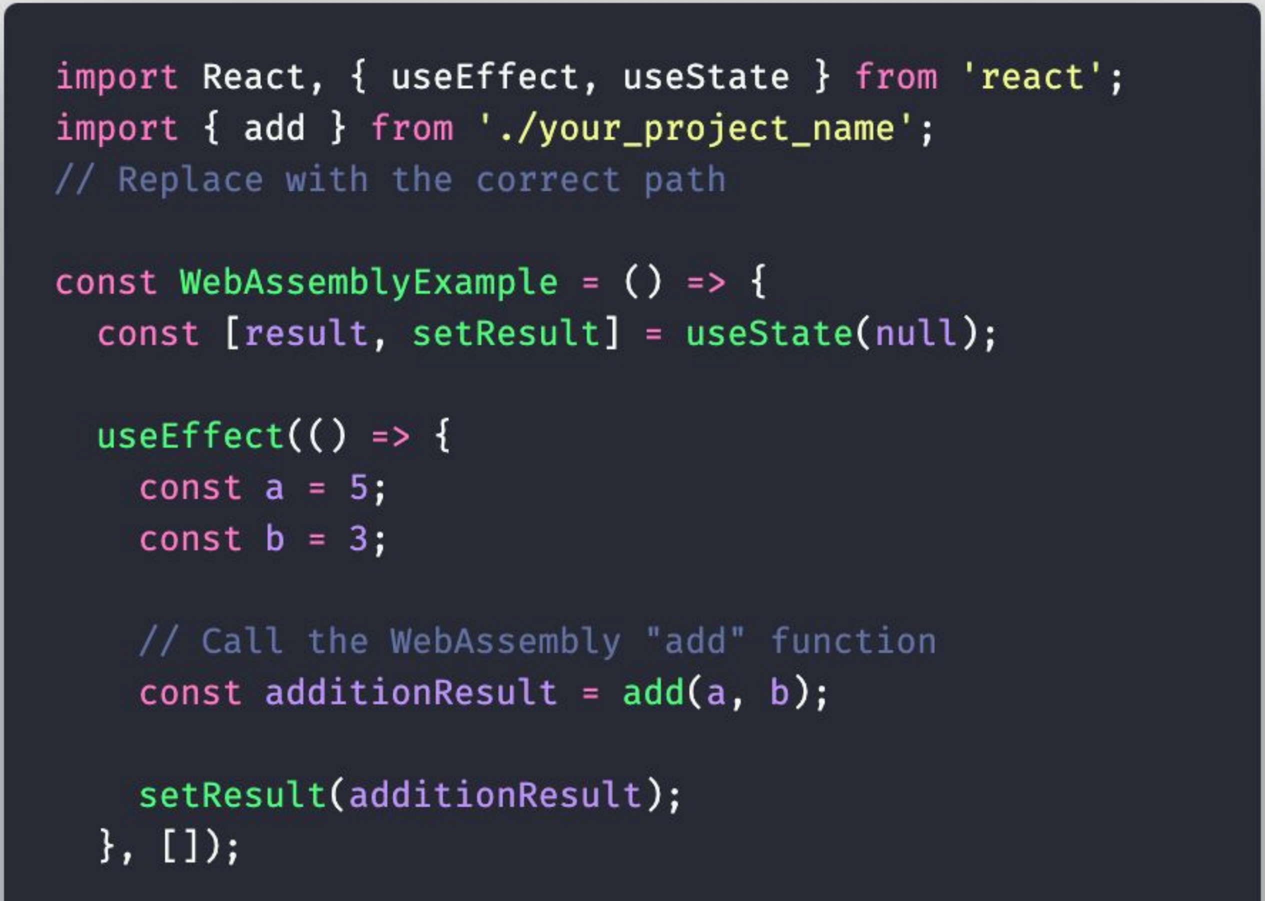
```rust
// lib.rs

use wasm_bindgen::prelude::*;

#[wasm_bindgen]
pub fn add(a: i32, b: i32) -> i32 {
    a + b
}
```



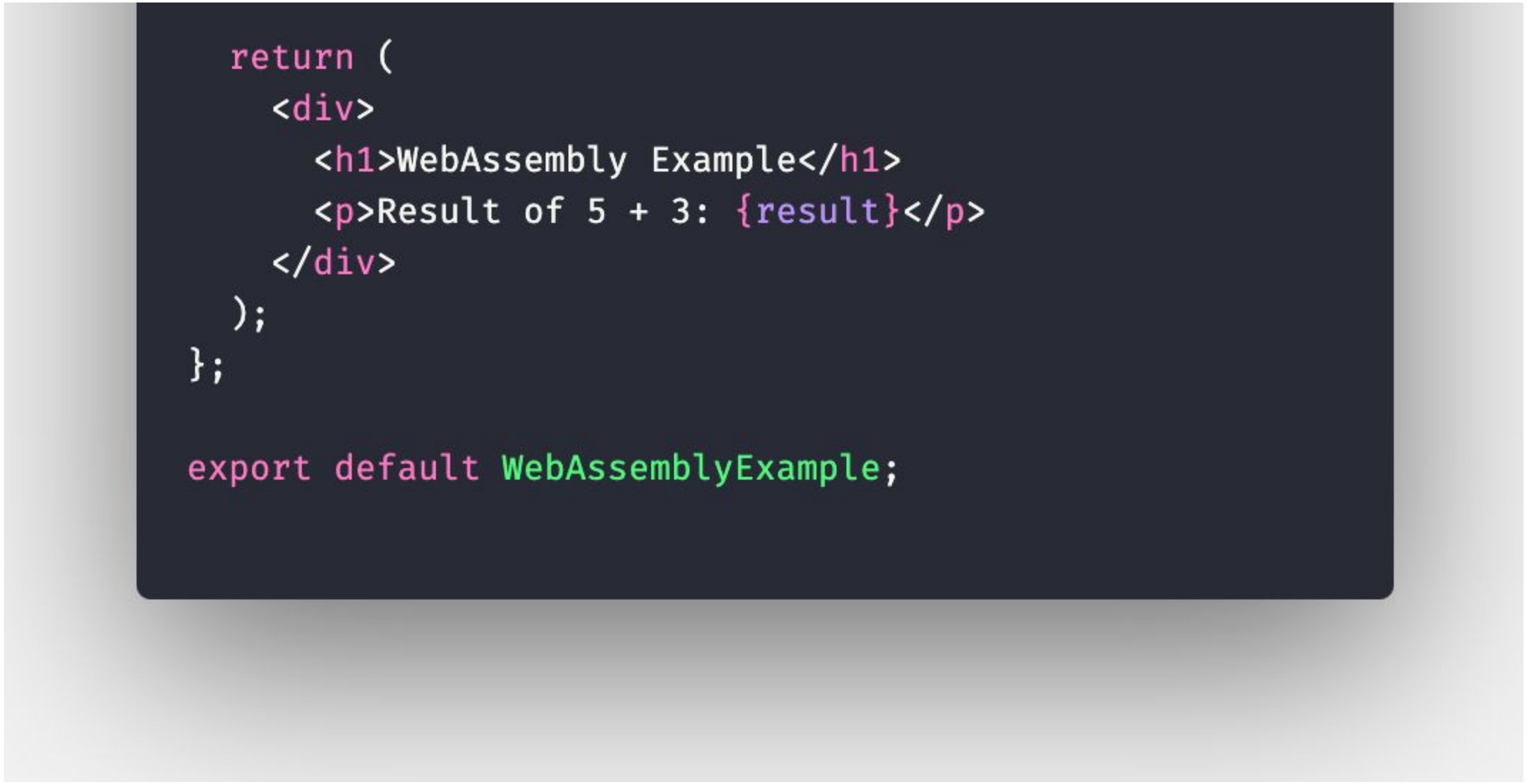
```toml
[lib]
crate-type = ["cdylib"]

[dependencies]
wasm-bindgen = "0.2"
```



```
wasm-pack build --target web
```



```
import React, { useEffect, useState } from 'react';
import { add } from './your_project_name';
// Replace with the correct path

const WebAssemblyExample = () => {
  const [result, setResult] = useState(null);

  useEffect(() => {
    const a = 5;
    const b = 3;

    // Call the WebAssembly "add" function
    const additionResult = add(a, b);

    setResult(additionResult);
  }, []);
```



```
  return (
    <div>
      <h1>WebAssembly Example</h1>
      <p>Result of 5 + 3: {result}</p>
    </div>
  );
};

export default WebAssemblyExample;
```

Of course, this is running in the browser, what about server-side?

It’s a server, an implementation detail!

https://ramonh.dev/react-wasm.pdf

Ramón Huidobro

Thank you, Friends!

ramonh.dev/card