WebAssembly auf der Serverseite

Pascal Fries

ATIX

was ist WebAssembly?

1996

1996

JavaScript 1.0

dynamische Websites

2009

2009

- frontend
- backend
- desktop
- …

2009

- frontend
- backend
- desktop
- ...

JS pros:

- sandbox
- nativ unterstützt im Browser
- portabel

JS cons:

- langsam
- Es ist JavaScript.

2017

WASM 1.0

Browser als compile Target

WASM pros:

- sandbox
- nativ unterstützt im Browser
- portabel
- schnell
- Es ist **nicht** JavaScript.

wie funktioniert WebAssembly?

1) Code in beliebiger Sprache schreiben
2) Kompilieren nach wasm32-x
3) Im Browser via JS instanziieren
4) ggf. exports aus JS aufrufen

Frontends „ohne“ JavaScript

- Problem: Wie „übersetzen“ zwischen DOM und WASM?

Frontends „ohne“ JavaScript

- Problem: Wie „übersetzen“ zwischen DOM und WASM?
- Lösung: `wasm-bindgen`

Frontends „ohne“ JavaScript

~~2008~~ 2023

- frontend
- backend
- desktop
- ...

WASM server warum und wie?

WASM 4 cloud:

- sicher (Sandbox)
- portabel (JITC Bytecode)
- schnell (kleiner Fußabdruck)

how 2 server:

1) WASM Laufzeitumgebung
   außerhalb des Browsers

2) ???

3) Profit

introducing

introducing

Wasmtime

Wasmer

WasmEdge

das Component Modell

die Idee

- WASM Runtimes lassen sich einfach in Anwendungen einbetten
- schnelle Instanziierung von Modulen
- nutzen als Format für Plugins

das Problem

- wie definiert man Interfaces?
- wie tauscht man Daten aus?

die Lösung

- eigenes Spec-Format für Interfaces: **WIT**
- kanonische ABI für Datenaustausch

was is WASI?

die Idee

nutze WIT, um Host-Interaktion zu definieren:

- Dateisystem
- Polls
- Sockets
- HTTP-Handler, KV-Stores, ...

der Stand

- Component-Modell ist recht weit
- Interface Types sind noch sehr früh

was mache ich jetzt damit?

fertige Runtimes

- Wasmtime
- Wasmer
- Wasmedge

unterstützen Filesystem, Sockets, ..., nach alter WASI Spec.

fertige Runtimes

- fermyon spin
- deislabs slight

unterstützen HTTP-Handler,
Messaging, DB, ..., mit Plugin-Modell

Build Your Own Runtime

Deployment

- direkt in der Runtime
- als „container“ (containerd-shims)

Zusammenfassung

- WASM fürs Frontend (+++)
- WASM als Plugin (++)
- WASI für Host-Interaktion (+)
- WASM Runtimes (++)

Danke!

Fragen?