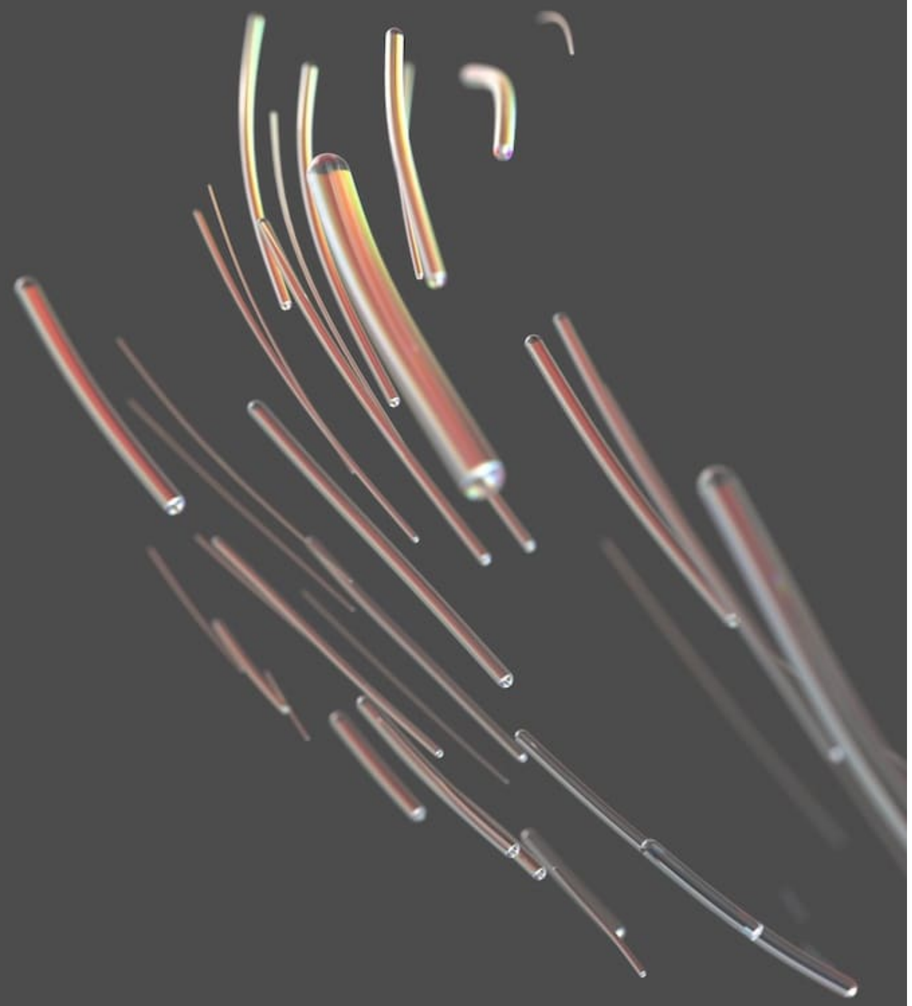


INNOQ Technology Day / Online / 2023-11-13

WebAssembly Security

INNOQ

CHRISTOPH ISERLOHN
SENIOR CONSULTANT

https://www.macports.org

Go Usergroup Rhein-Ruhr

https://go-rr.org

28.08.2023
Passkeys

Nutzbar und sicher?

→

CHRISTOPH ISERLOHN
SONJA SCHEUNGRAB

26.06.2023
Browser Security

Was hat der Browser je für uns getan?

→

LISA MARIA MORITZ
CHRISTOPH ISERL

11.04.2023
KI und Security

Eine Risikobewertung

→

CHRISTOPH ISERLOHN
FELIX SCHUMACHER

13.03.2023
Firewall

Die Wiege der falschen Sicherheit

→

LISA MARIA MORITZ

CHRISTOPH ISERLOHN

26.01.2023
Das LastPass Drama

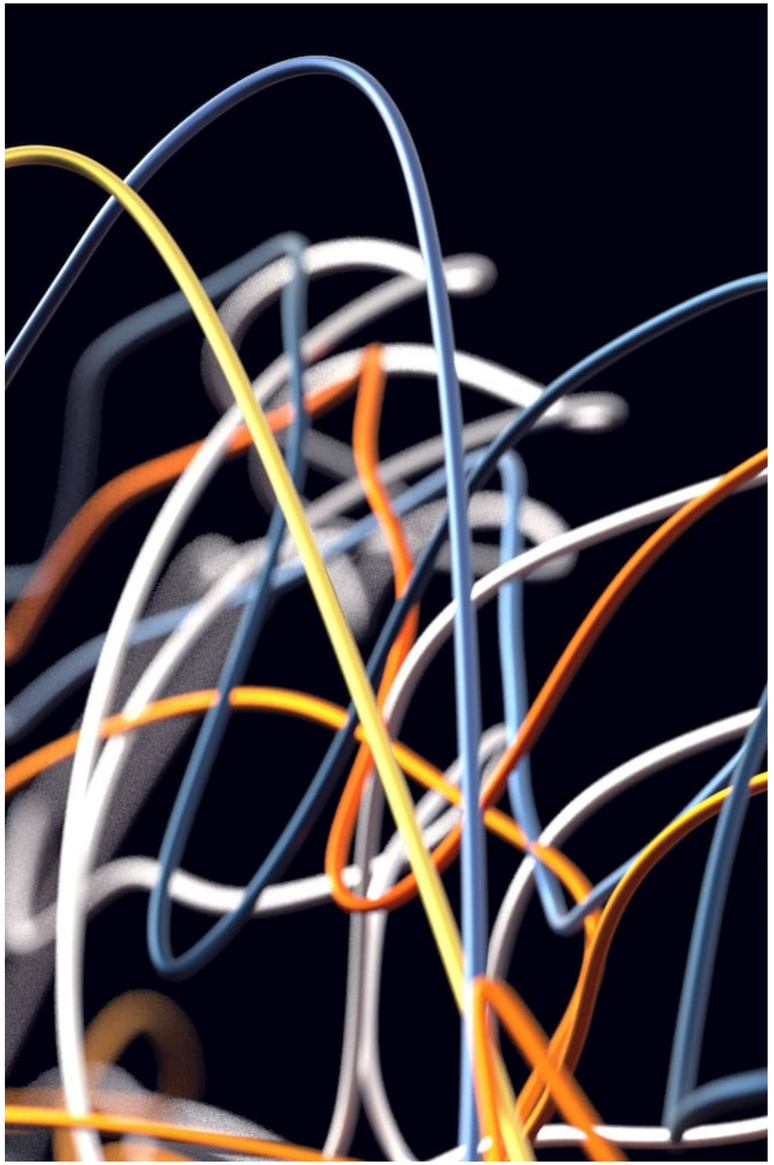
Passwort-Manager? Na klar. Aber doch nicht so!

→

LISA MARIA MORITZ
CHRISTOPH ISERLOHN

Agenda

- Ein kurzer Blick in die Geschichte
- WebAssembly im Browser
- WebAssembly auf dem Server
- Plugins mit WebAssembly
- Sandboxing mit WebAssembly
- Quo Vadis, WebAssembly



Asm.js

Ein kurzer Blick in die Geschichte


mozilla
EPIC
GAMES

C

```c
int inc(int n) {
  return n + 1;
}
```

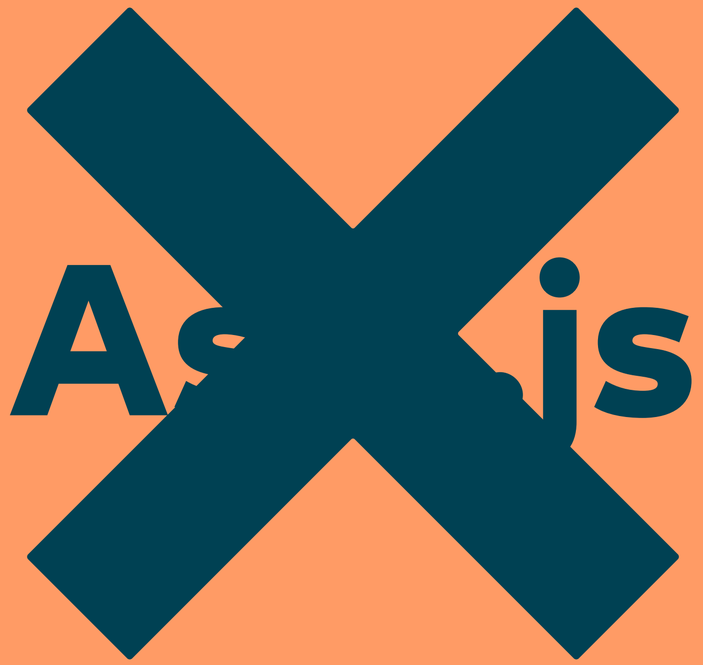
JavaScript

```
function inc(n) {
  return (n + 1);
}
```

Asm.js

```
function inc(n) {
  n = n|0;
  return (n + 1)|0;
}
```

Emscripten


Axis


WA

WebAssembly Working Group

The mission of the WebAssembly Working Group is to standardize a size- and load-time-efficient format and execution environment, allowing compilation to the web with consistent behavior across a variety of implementations.

Homepage This page

Status Active

Charter Chartered until 31 July 2022 (history)

Shortname wasm

Summary


W3C
COMMUNITY & BUSINESS GROUPS
Home / WebAssembly Community Group
WEBASSEMBLY COMMUNITY GROUP
The mission of this group is to promote early-stage cross-browser collaboration on a new, portable, size- and load-time-efficient format suitable for compilation to the web.

WebAssembly - die Kurzfassung

- Stack basierte VM
- Binäres Format
- Sicher
- Offen

WebAssembly - Use Cases

- Audio/Video-Anwendungen
- AAA-Spiele
- VR / AR
- ...

C

```c
int add(int x, int y) {
  return x + y;
}
```

WAT

```
(module
    (func $add (param $0 i32) (param $1 i32) (result i32)
        (i32.add
            (local.get $0)
            (local.get $1)
        )
    )
    (export "add" (func $add))
)
```

JavaScript / WebAssembly

```
const binary = require('fs').readFileSync('add.wasm');

WebAssembly.instantiate(binary).then(({ instance }) => {
  console.log(instance.exports.add(40, 2));
});
```

Was hat das ganze mit Security zu tun?

Solomon Hykes / @shykes@hachyderm.io
@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!

Post übersetzen

> **Lin Clark** @linclark · 27. März 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...

9:39 nachm. · 27. März 2019

WASI

The WebAssembly System Interface

WASI is a modular system interface for WebAssembly. As described in the initial announcement, it's focused on security and portability.

WASI is being standardized in a subgroup of the WebAssembly CG. Discussions happen in GitHub issues, pull requests, and bi-weekly Zoom meetings.

For a quick intro to WASI, including getting started using it, see the intro document.

The Wasmtime runtime's tutorial contains examples for how to target WASI from C and Rust. The resulting .wasm modules can be run in any WASI-compliant runtime.

For more documentation, see the documents guide.


WASI
wasi-preview1
X
wasmer-wasi

BYTECODE
ALLIANCE

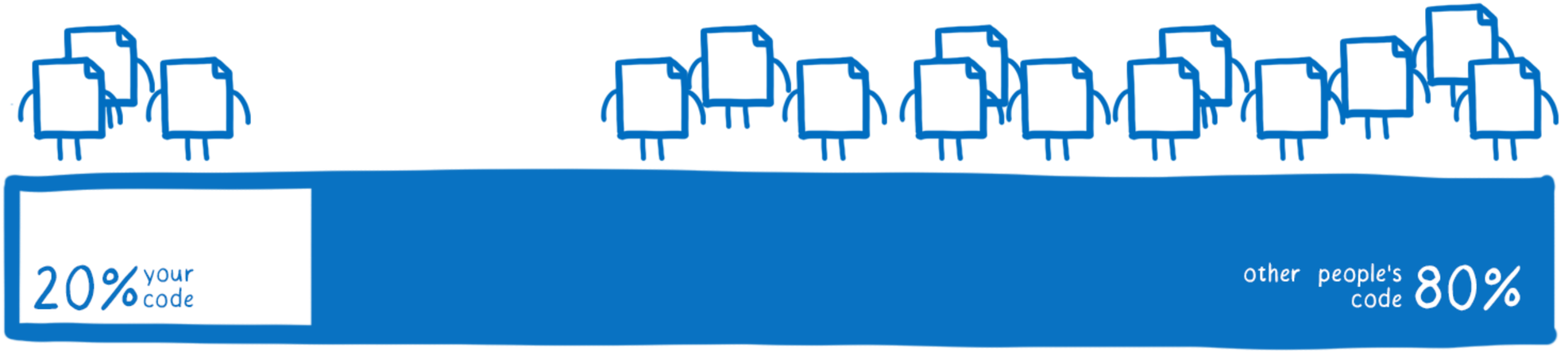
Das Problem

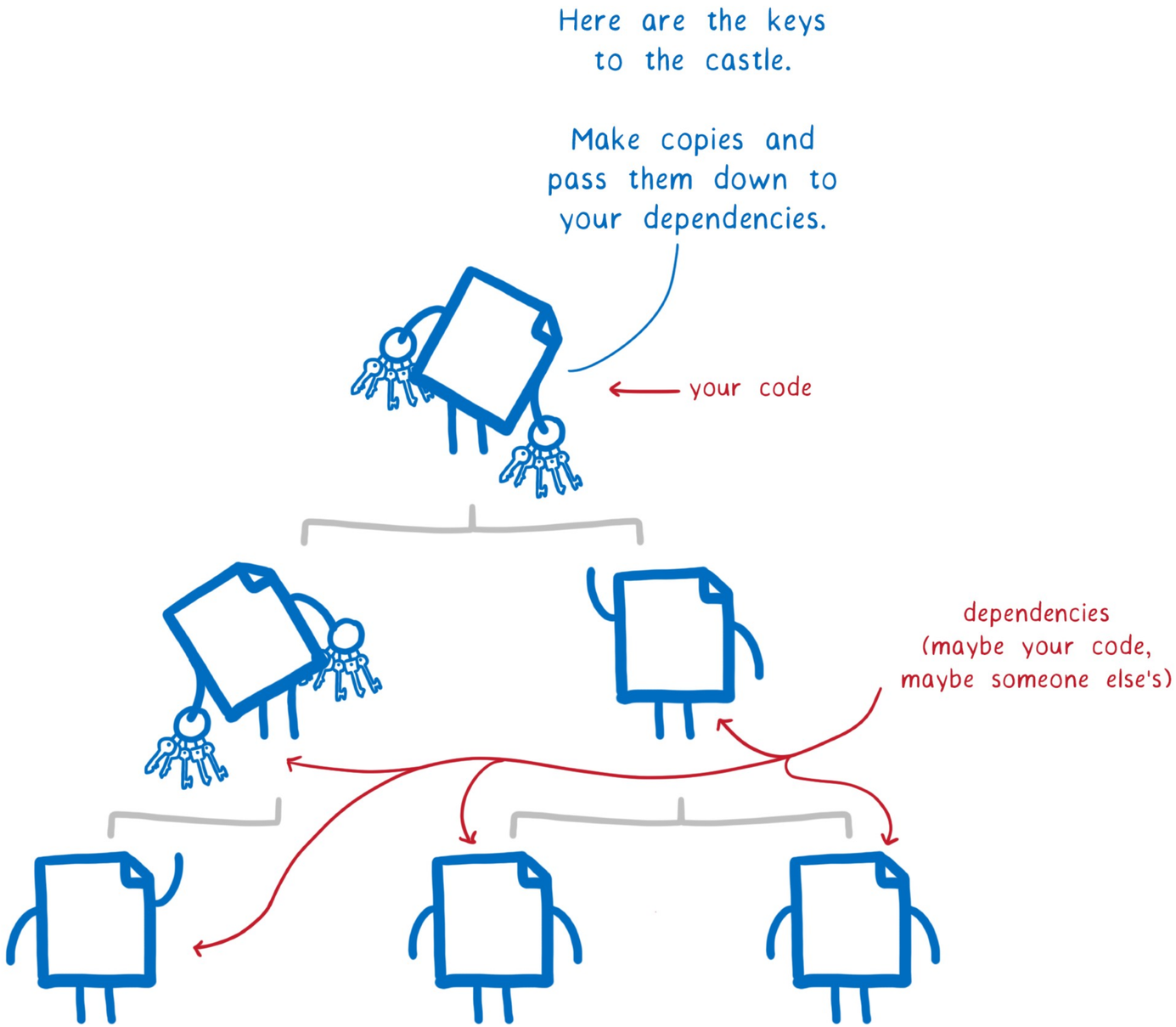
„This is a unique moment in time at the dawn of a new technology, where we have the opportunity to fix what's broken and build new, **secure-by-default foundations** for native development that are portable and scalable."

Luke Wagner,

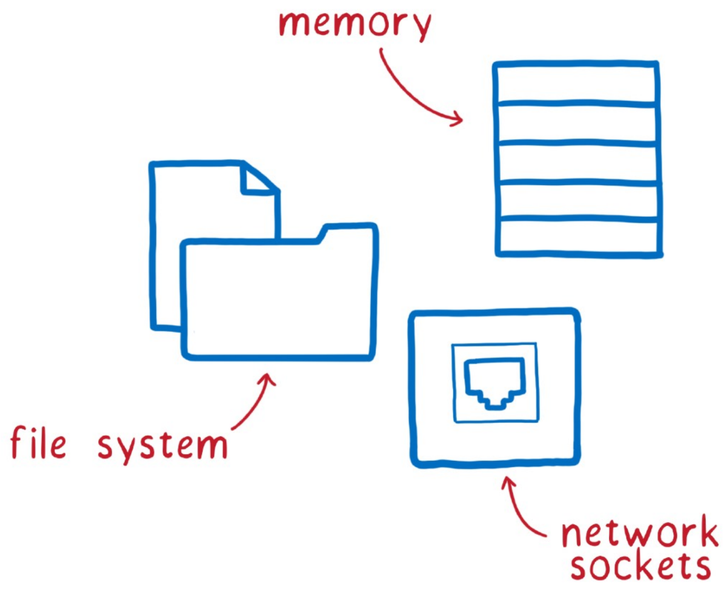
Distinguished Engineer at Mozilla and co-creator of WebAssembly

composition of an average code base

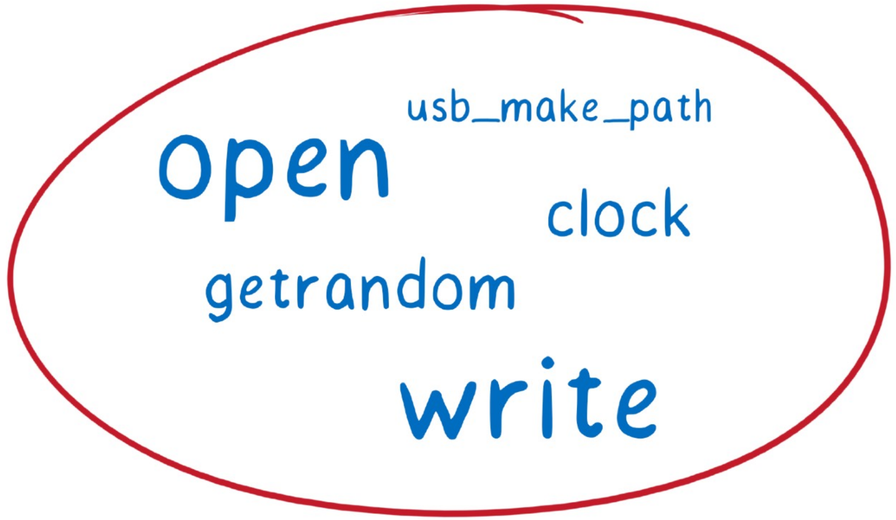


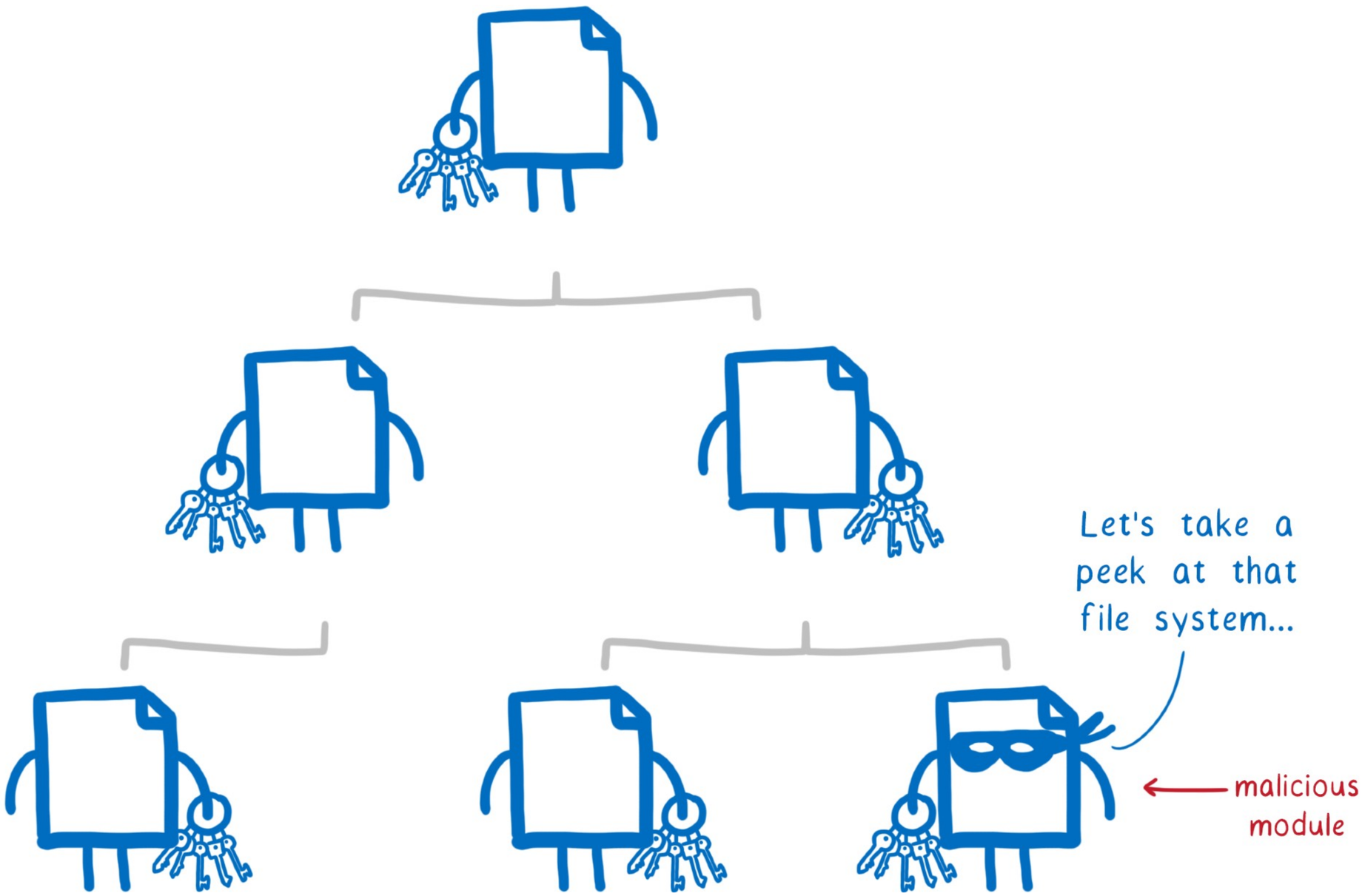


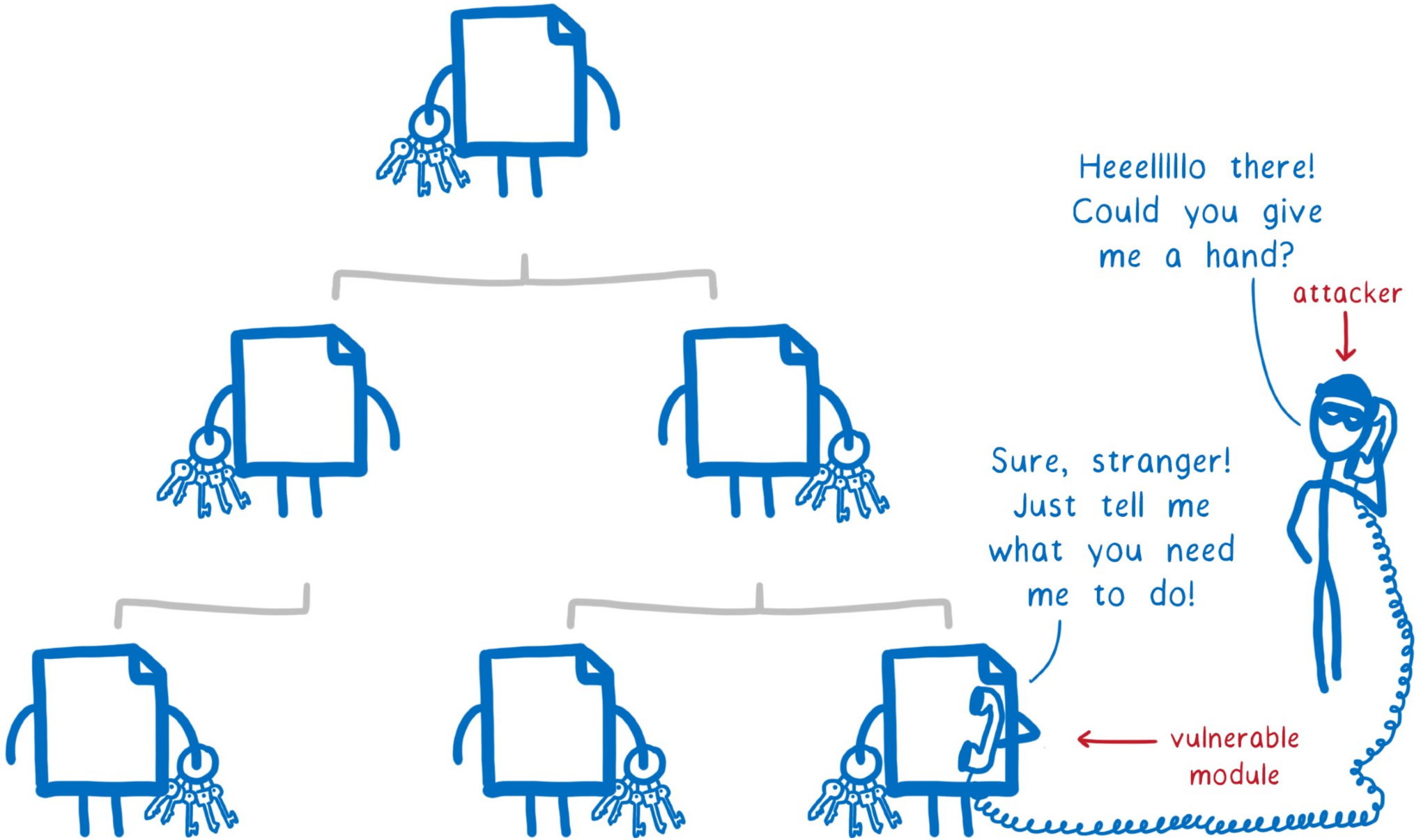
system resources



host-provided APIs
& syscalls







Was tun wir heute?

Manueller Code-Review

Tooling

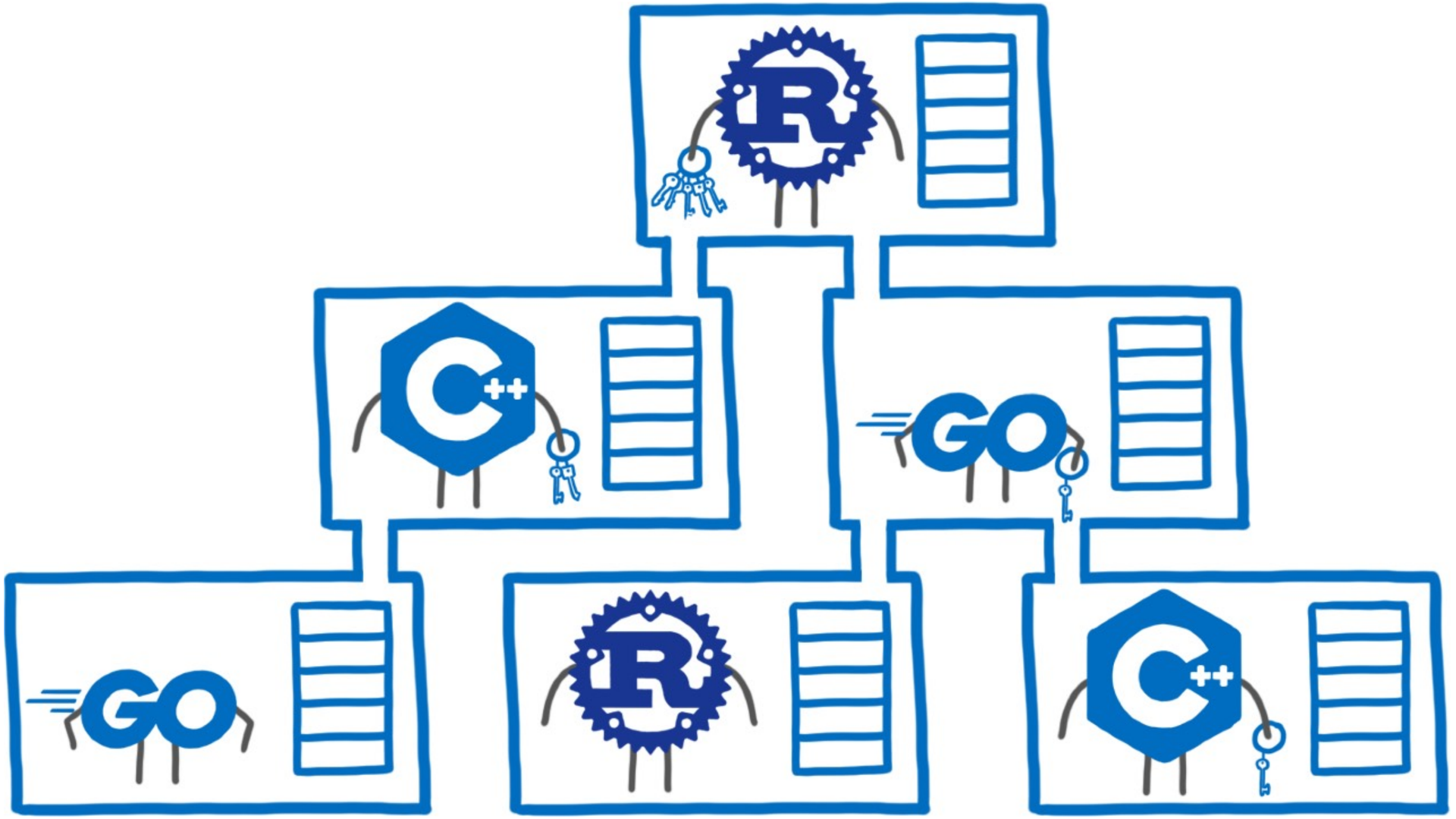
Vertrauen

¯_(ツ)_/¯

Wie kann WebAssembly Abhilfe schaffen?

Security Features

- Statisch typisiert
- Control-Flow-Integrity
- Geschützter Call-Stack
- Für Sandboxing ausgelegt
- Principle of Least Authority / Capabilities


GO
GO

Gibt es das schon irgendwo?


KRUSTLET

Why Containers and WebAssembly Work Well Together

Tyler Charboneau

Editor's note: The Docker+Wasm Technical Preview is now available. Find out more about the preview and try it for yourself!

Posted

Jul 1, 2022

Post Tags

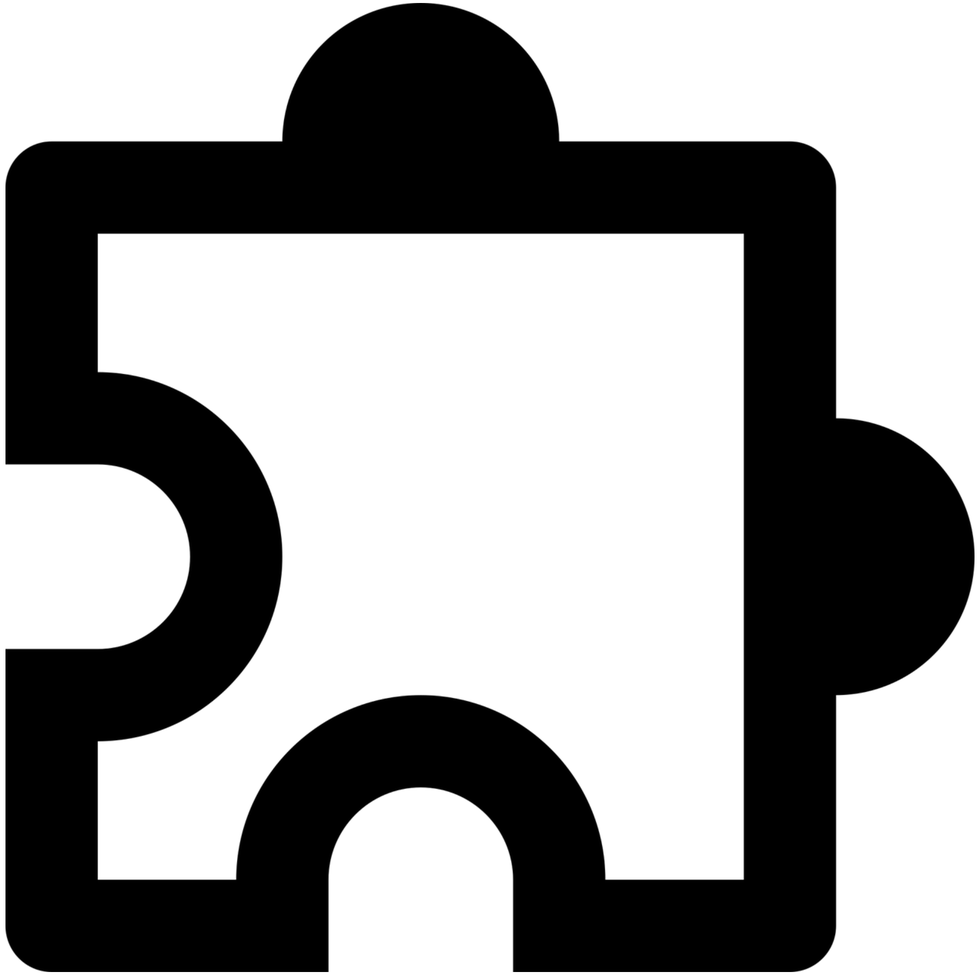
- containers
- Docker Desktop
- Docker Hub
- Wasm
- WebAssembly


fastly
®


cosmonic


wasmcloud

Und ohne „Cloud“?



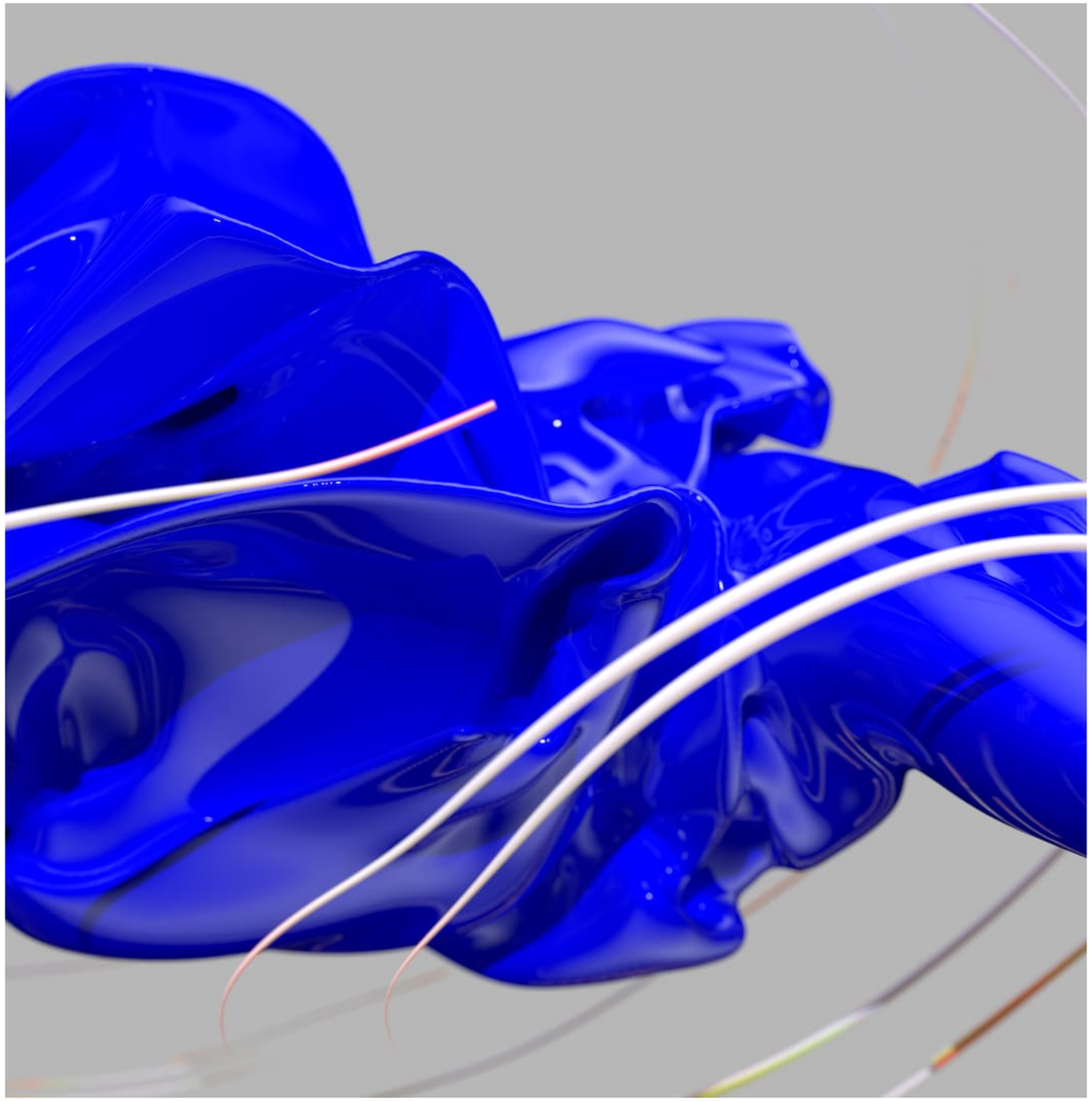
RLBox

WebAssembly and Back Again: Fine-Grained Sandboxing in Firefox 95

By Bobby Holley

Posted on December 6, 2021 in Featured Article, Firefox, and JavaScript

Womit kann ich WebAssembly nutzen?



Sprachen

- C/C++
- Rust
- Go
- Zig
- ...

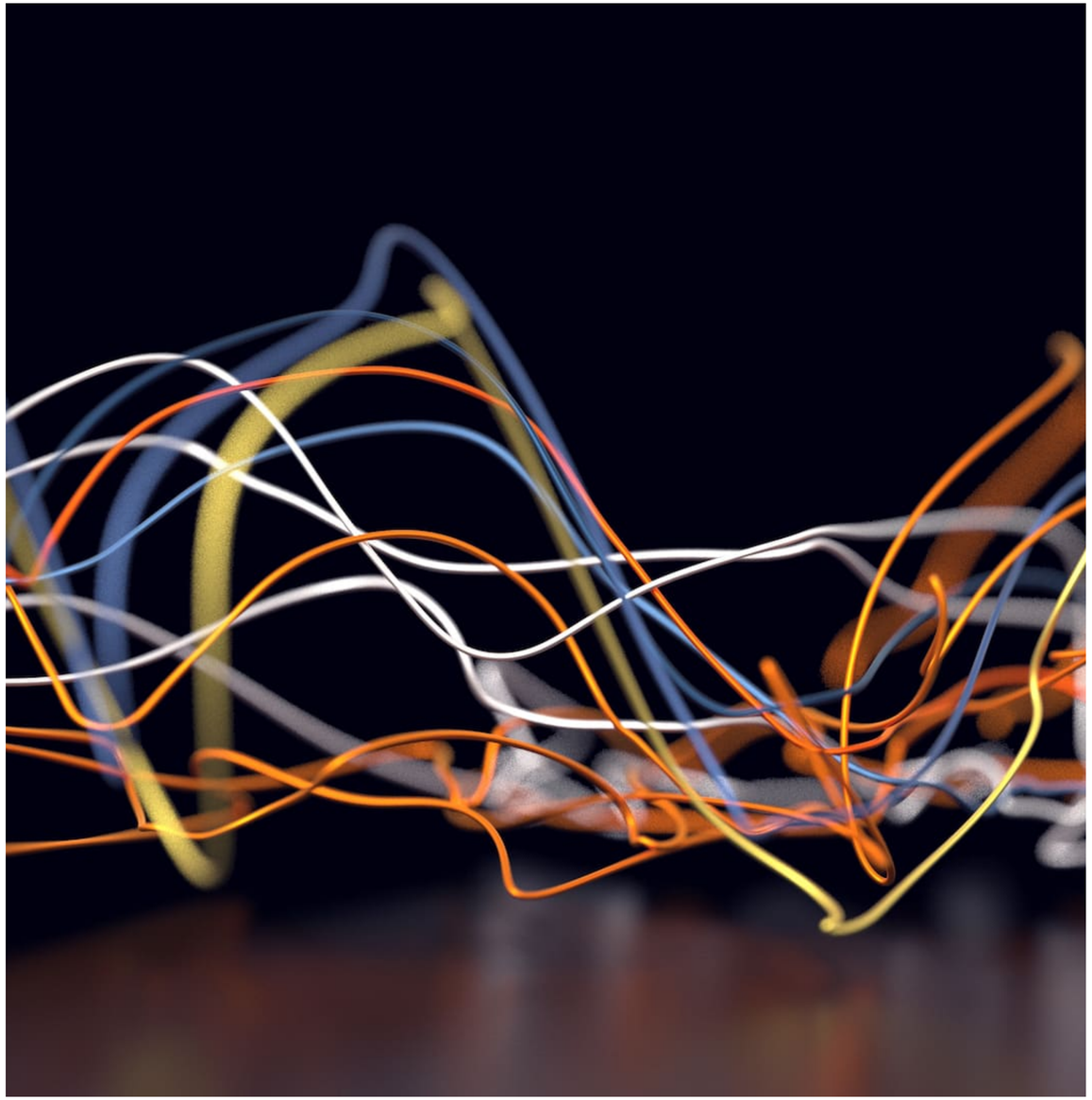
Runtimes

Bytecode alliance:

- Wasmtime (Rust)
- WebAssembly Micro Runtime (C)

3rd-Party:

- Wasmer (Rust)
- Waszero (Go)
- Wasmedge (C++)



Quo Vadis, WebAssembly?


Answers 1km →


Java

Danke! Fragen?

Christoph Iserlohn
christoph.iserlohn@innoq.com
@ci@innoq.social

INNOQ
www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Hermannstrasse 13
20095 Hamburg

Erftstr. 15-17
50672 Köln

Königstorgraben 11
90402 Nürnberg