

# Using WebAssembly to run, extend, and secure your .NET application

# Niels Tanis



# Update Conference Prague 2022

0101  
0101

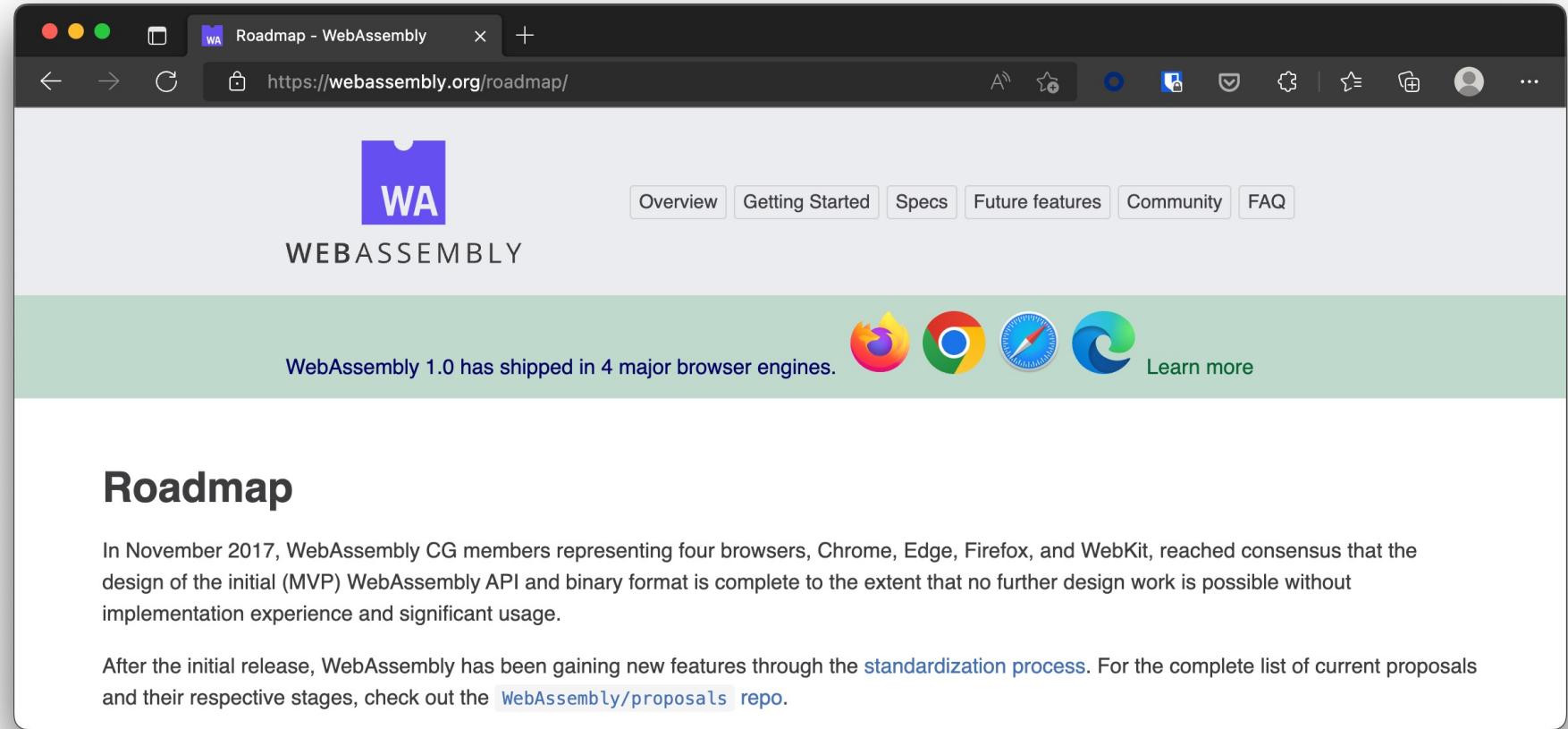
# Who am I?

- Niels Tanis
- Sr. Principal Security Researcher @ Veracode
  - Background .NET Development,  
Pentesting/ethical hacking,  
and software security consultancy
  - Research on static analysis for .NET apps



# WebAssembly

0101  
0101



A screenshot of a web browser window showing the 'Roadmap - WebAssembly' page at <https://webassembly.org/roadmap/>. The page features a large 'WA' logo and the word 'WEBASSEMBLY'. A green banner at the top states 'WebAssembly 1.0 has shipped in 4 major browser engines.' followed by icons for Firefox, Chrome, Edge, and Safari, and a 'Learn more' link. Below the banner, a section titled 'Roadmap' contains text about the initial consensus reached in November 2017 and links to the standardization process and proposals repository.

Roadmap

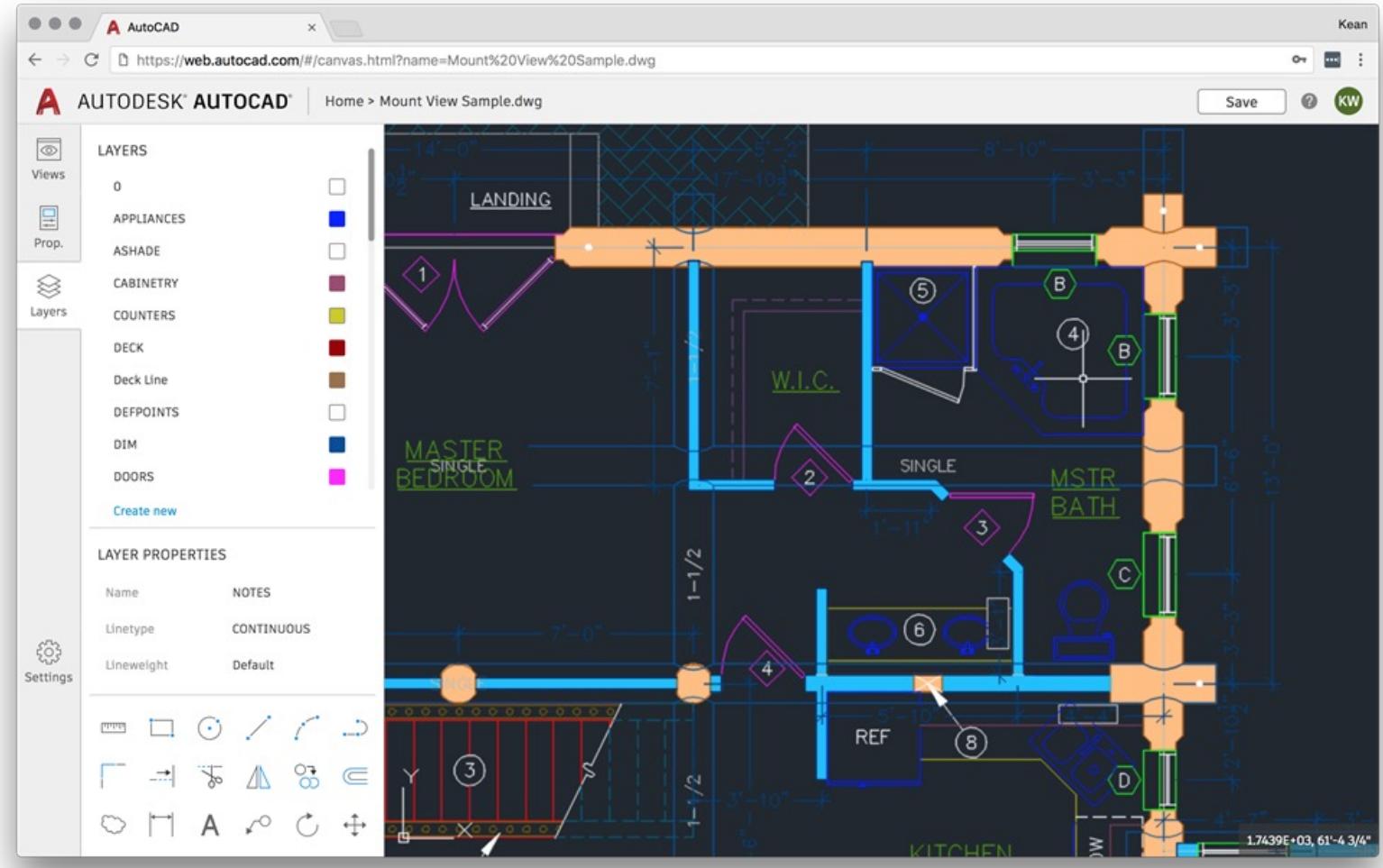
In November 2017, WebAssembly CG members representing four browsers, Chrome, Edge, Firefox, and WebKit, reached consensus that the design of the initial (MVP) WebAssembly API and binary format is complete to the extent that no further design work is possible without implementation experience and significant usage.

After the initial release, WebAssembly has been gaining new features through the [standardization process](#). For the complete list of current proposals and their respective stages, check out the [WebAssembly/proposals](#) repo.



0101  
0101

# WebAssembly - AutoCAD



# WebAssembly - SDK's

0101  
0101

A screenshot of a Medium article page. The title is "Introducing the Disney+ Application Development Kit (ADK)". It's published in the "disney-streaming" section. The author is Mike Hanley, with a profile picture and a "Follow" button. The article was published on Sep 8, 2021, and has a 10 min read time. There are social sharing icons for Twitter, Facebook, LinkedIn, and a "Get started" button. Below the title, there's a short bio for Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices. The page includes a navigation bar with icons for home, search, and user profile.

A screenshot of a Medium article page. The title is "How Prime Video updates its app for more than 8,000 device types". It's published in the "CLOUD AND SYSTEMS" section under the "amazon | science" header. The author is Alexandru Ene, with a profile picture and a "Subscribe" button. The article was published on January 27, 2022. A text snippet says, "The switch to WebAssembly increases stability, speed." Below the article, there's a footer note: "At [Prime Video](#), we're delivering content to millions of customers on".



0101  
0101

# Agenda

- Introduction
- WebAssembly 101
- Running .NET on WebAssembly
- Extending .NET with WebAssembly
- Securing .NET with WebAssembly
- Conclusion
- Q&A





# WebAssembly Design

- **Be fast, efficient, and portable**
  - Executed in near-native speed across different platforms
- **Be readable and debuggable**
  - In low-level bytecode but also human readable
- **Keep secure**
  - Run on sandboxed execution environment
- **Don't break the web**
  - Ensure backwards compatibility



0101  
0101

# WebAssembly

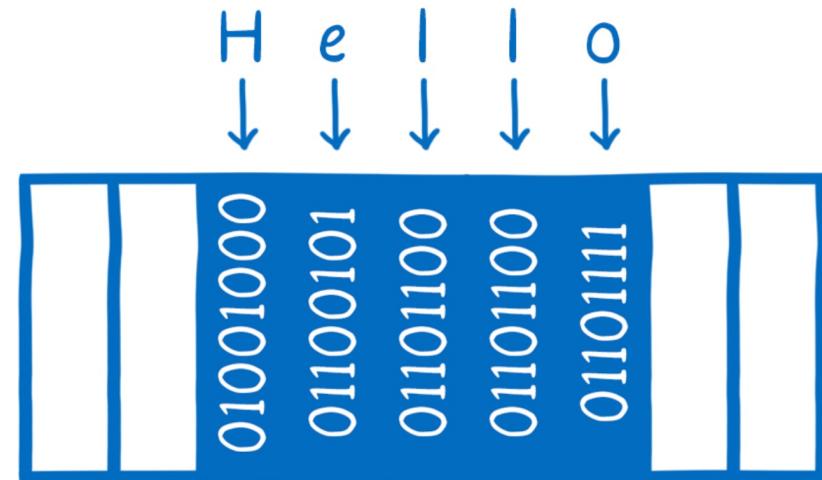
- Binary instruction format for stack-based virtual machine similar to .NET running MSIL
- Designed as a portable compilation target
- The security model of WebAssembly:
  - Protect users from buggy or malicious modules
  - Provide developers with useful primitives and mitigations for developing safe applications

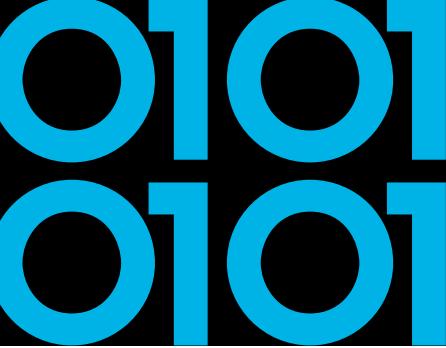


0101  
0101

# WebAssembly Memory

- Isolated per WASM module
- A contiguous, mutable array of uninterpreted bytes





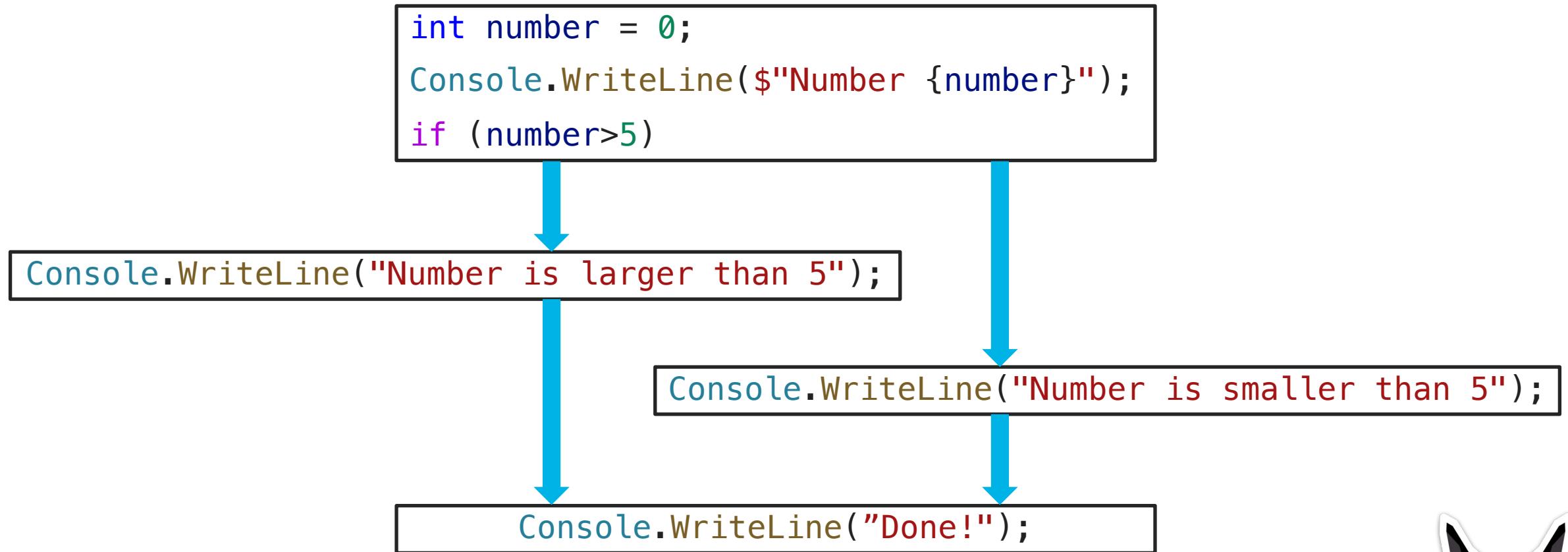
# WebAssembly Control-Flow Integrity

```
int number = 0;  
Console.WriteLine($"Number {number}");  
if (number>5)  
{  
    Console.WriteLine("Number is larger than 5");  
}  
else  
{  
    Console.WriteLine("Number is smaller than 5");  
}  
Console.WriteLine("Done!");
```



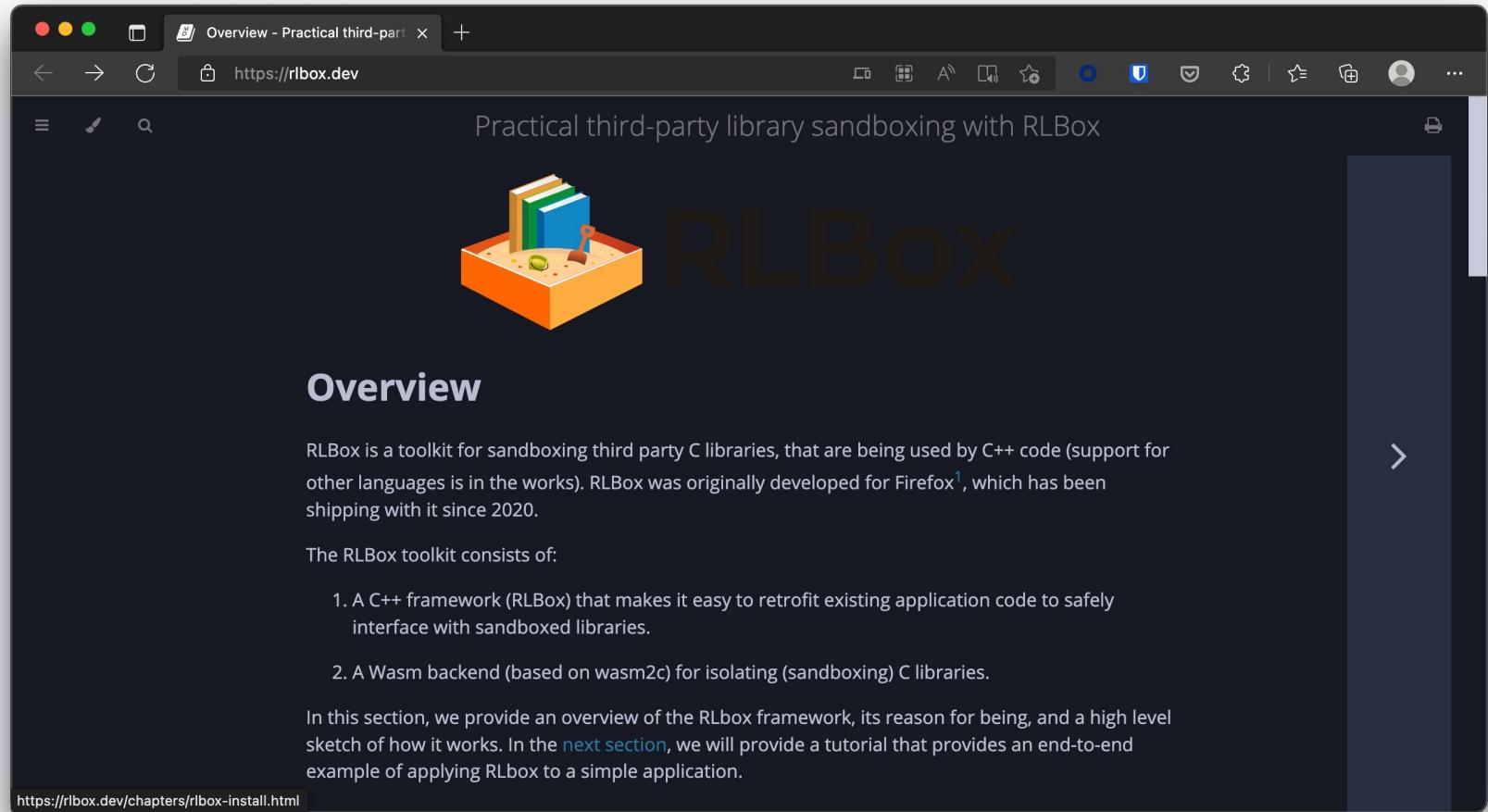
0101  
0101

# WebAssembly Control-Flow Integrity



0101  
0101

# FireFox RLBox



Overview - Practical third-party library sandboxing with RLBox

https://rlbox.dev

## Overview

RLBox is a toolkit for sandboxing third party C libraries, that are being used by C++ code (support for other languages is in the works). RLBox was originally developed for Firefox<sup>1</sup>, which has been shipping with it since 2020.

The RLBox toolkit consists of:

1. A C++ framework (RLBox) that makes it easy to retrofit existing application code to safely interface with sandboxed libraries.
2. A Wasm backend (based on wasm2c) for isolating (sandboxing) C libraries.

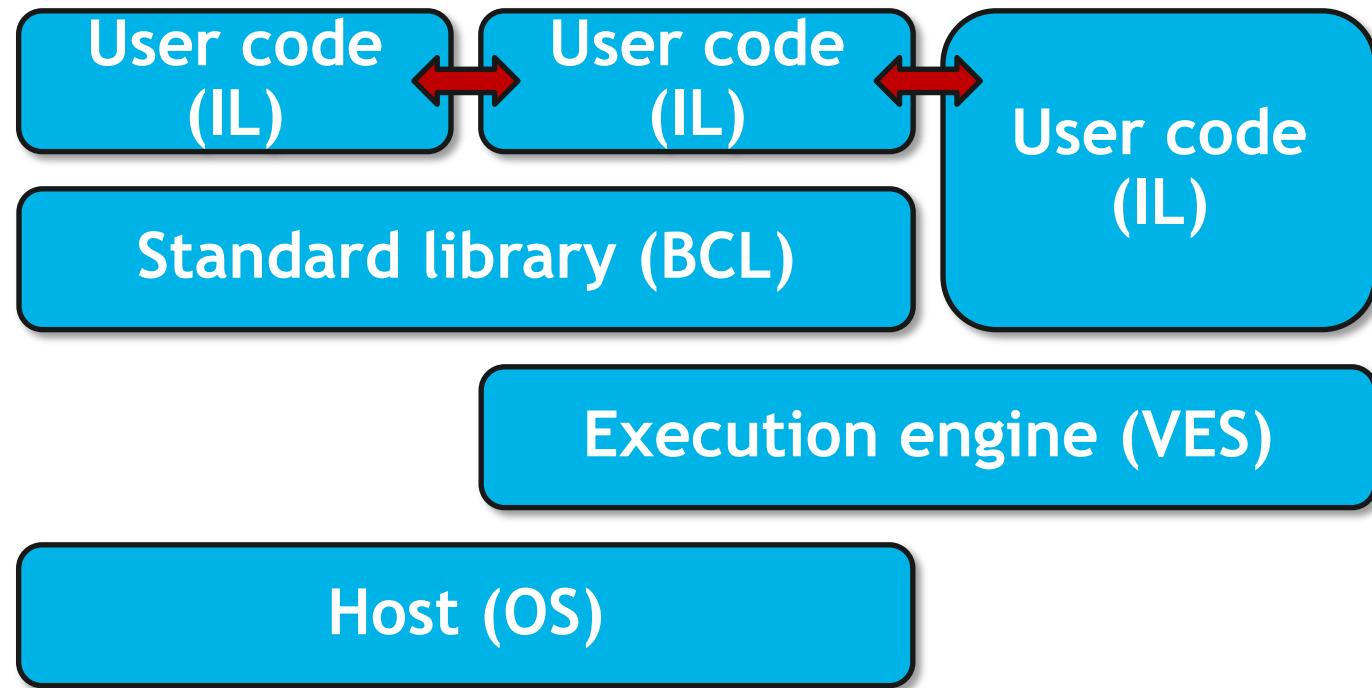
In this section, we provide an overview of the RLbox framework, its reason for being, and a high level sketch of how it works. In the [next section](#), we will provide a tutorial that provides an end-to-end example of applying RLbox to a simple application.

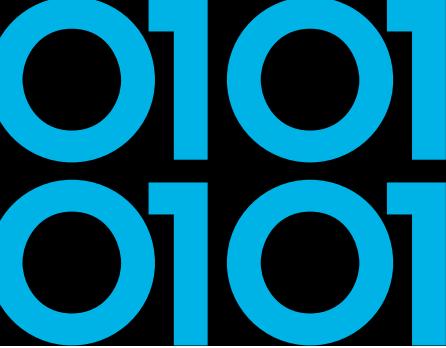
<https://rlbox.dev/chapters/rbbox-install.html>



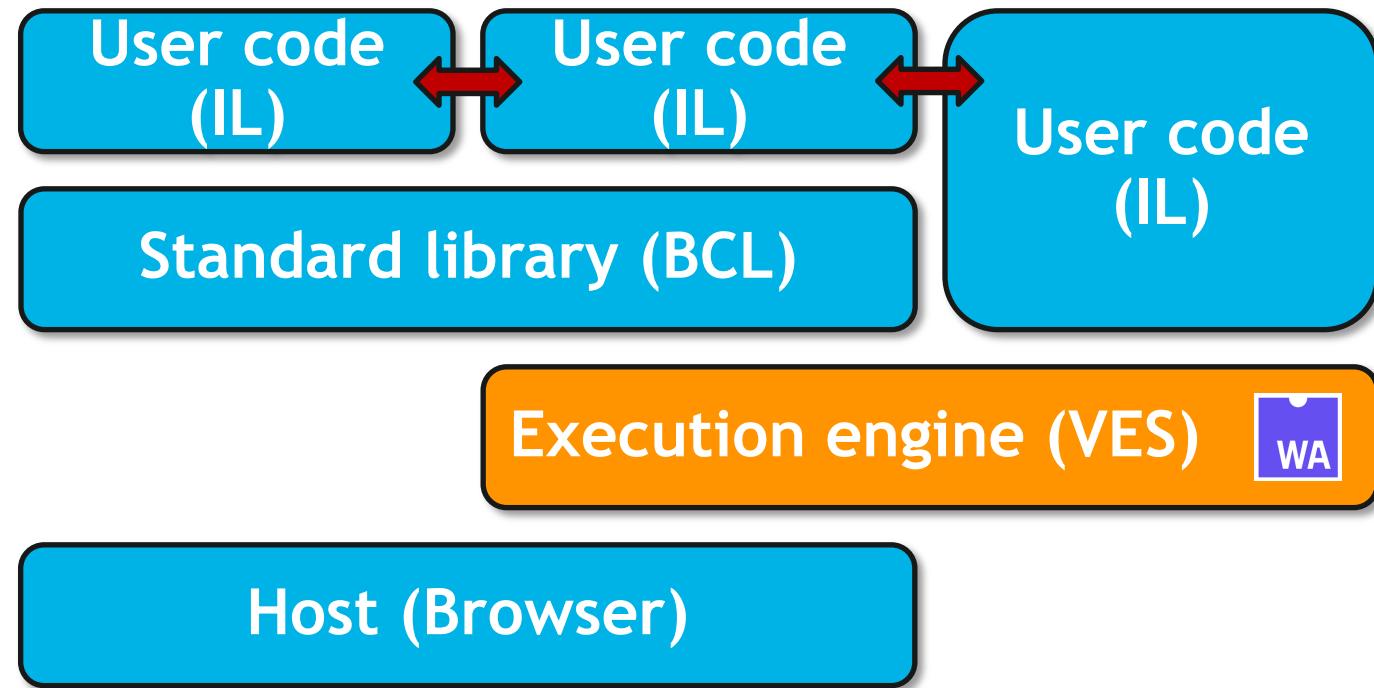


# Running .NET on WebAssembly



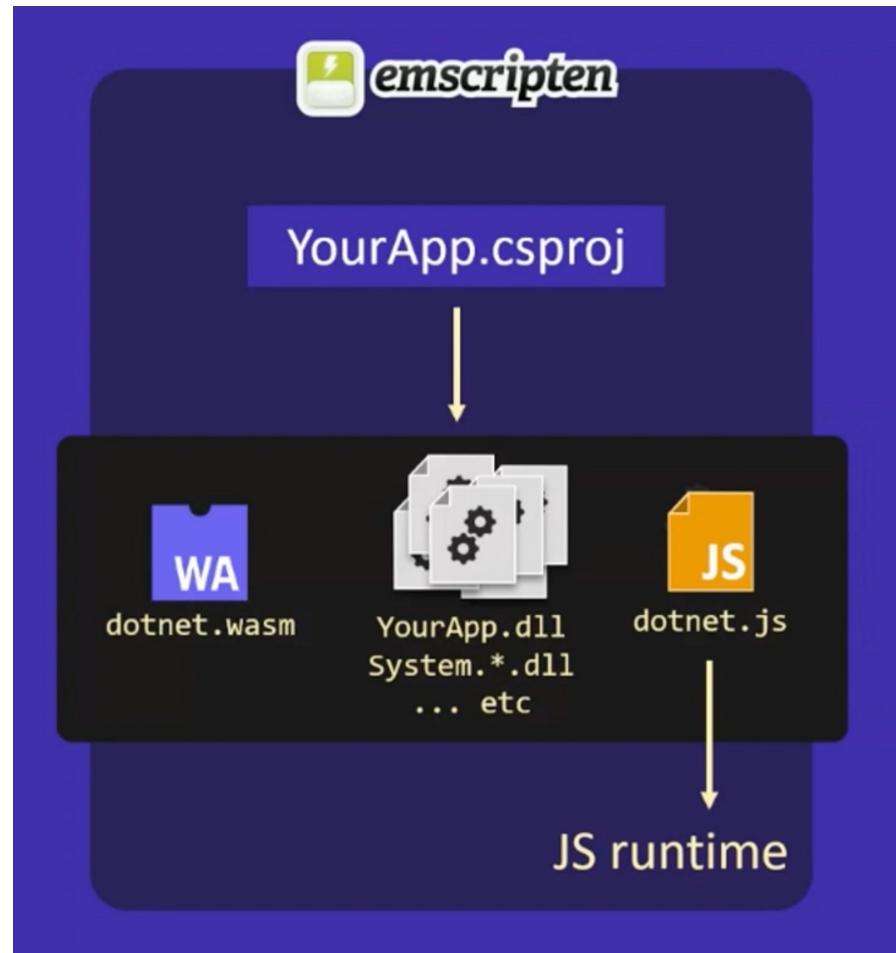


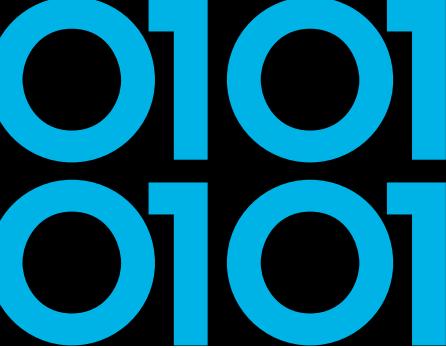
# Running .NET on WebAssembly



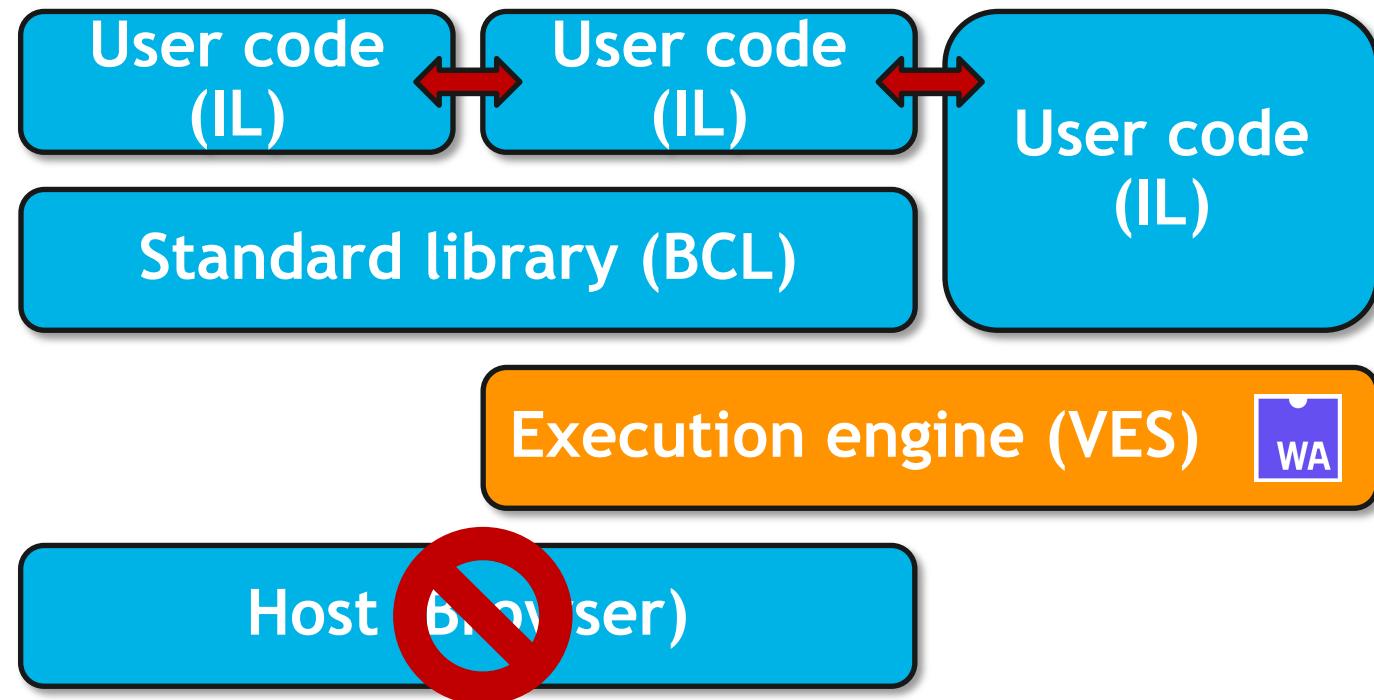
0101  
0101

# Blazor WebAssembly





# Running .NET on WebAssembly



# WebAssembly System Interface WASI



- Introduced in March 2019 by Bytecode Alliance
- WasmTime implementation as reference
- POSIX inspired, engine-independent, non-Web system-oriented API for WebAssembly

# WebAssembly System Interface WASI



- Strong sandbox with Capability Based Security
- Right now, supports FileSystem actions
- Future support for sockets and other system resources.
- Anyone recall .NET Standard? ☺

0101  
0101

# Docker vs WASM & WASI

A screenshot of a Twitter web client window. The header shows the URL <https://twitter.com/s...>. The main content is a tweet from **Solomon Hykes** (@solomonstre) titled "Collectie". The tweet text is:

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!

Below the tweet, there is a reply from **Lin Clark** (@linclark) dated March 27, 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...](https://hacks.mozilla.org/2019/03/standa...)

[Deze collectie weergeven](#)

The timestamp at the bottom is 9:39 p.m. · 27 mrt. 2019 · Twitter Web Client.



0101  
0101

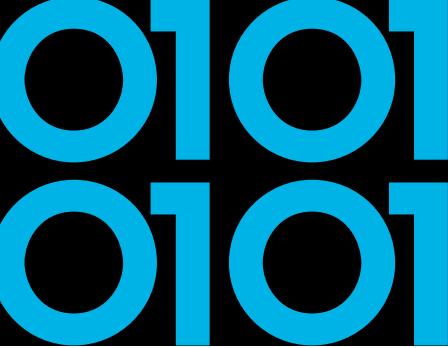
# Docker vs WASM & WASI

A screenshot of a Twitter web browser window. The tweet is from Solomon Hykes (@solomonstre) and reads: "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 :)

The tweet was posted at 4:50 a.m. on March 28, 2019, via the Twitter Web App. It has 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.



# Docker & WASM

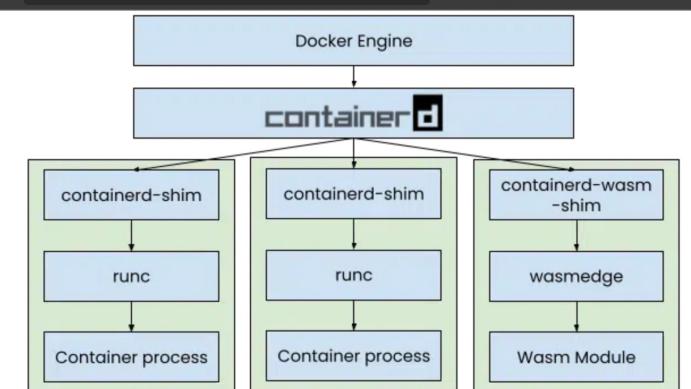


Introducing the Docker+Wasm Technical Preview

 MICHAEL IRWIN

Oct 24 2022

The [Technical Preview of Docker+Wasm](#) is now available! Wasm has been producing a lot of buzz recently, and this feature will make it easier for you to quickly build applications targeting Wasm runtimes.



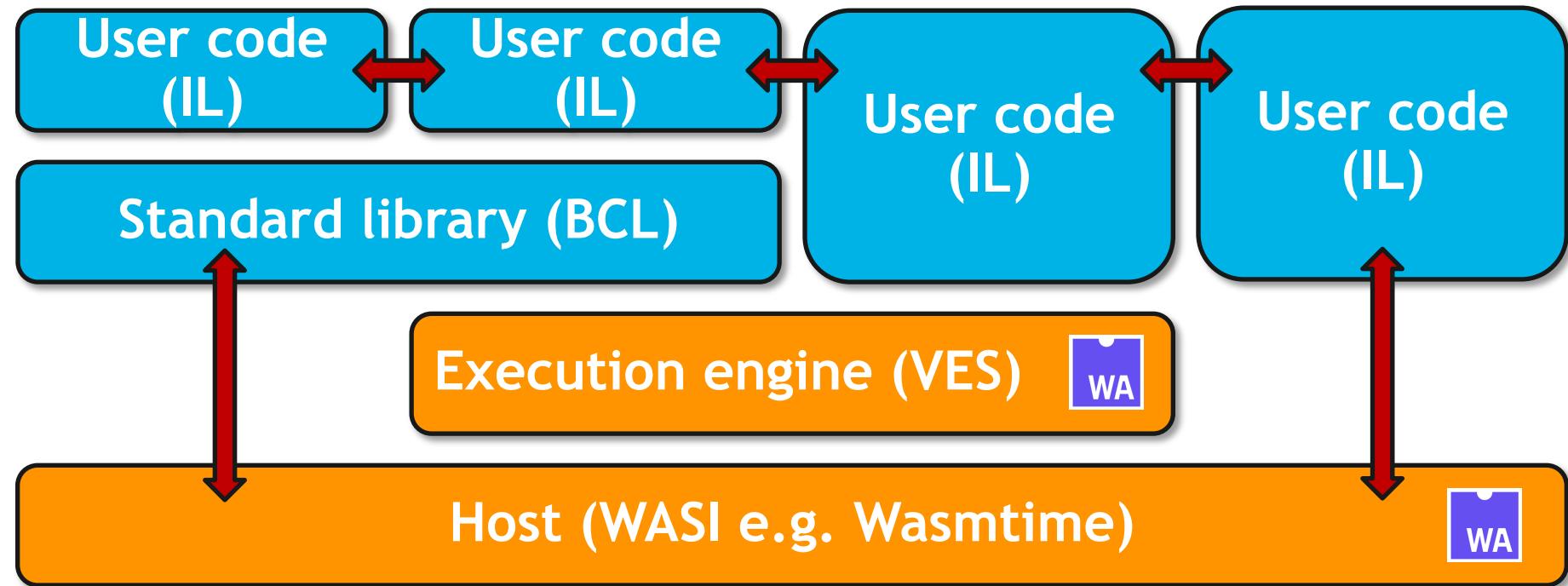
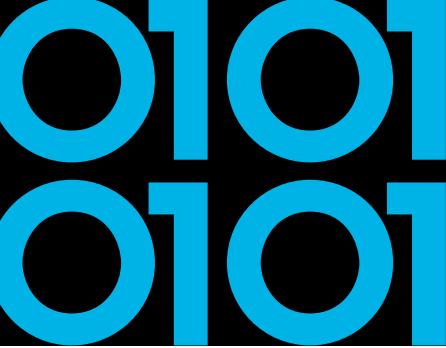
Let's look at an example!

After installing the preview, we can run the following command to start an example Wasm application:

```
docker run -dp 8080:8080 --name=wasm-example --runtime=io.containerd.wasmedge.v1 --platform=wasi/wasm32 michaelirwin244/wasm-example
```



# WebAssembly System Interface WASI



0101  
0101

# Experimental WASI SDK for .NET



0101  
0101

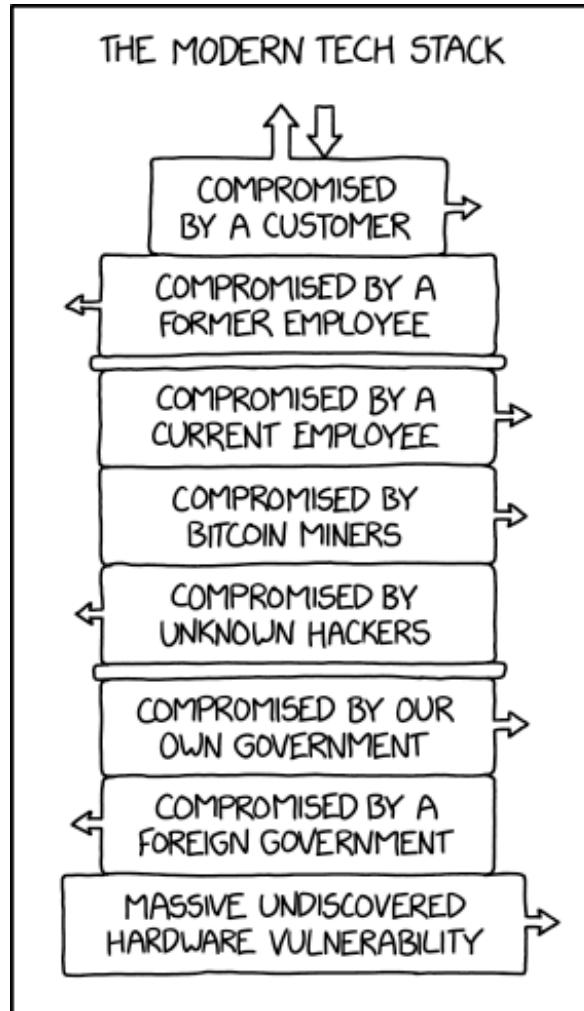
# Extending .NET with WASM

- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!



0101  
0101

# Trusted Computing - XKCD 2166



0101  
0101

# Enarx

The screenshot shows a dark-themed web browser window displaying the Enarx homepage at <https://enarx.dev>. The page features a large white "Enarx" logo at the top center, followed by the tagline "Confidential Computing with WebAssembly". Below the tagline are two buttons: a green "Download" button and a blue "Try Enarx" button. A large circular icon resembling a stylized keyhole or lock is centered on the page. At the bottom, the text "100% Open Source" is displayed above a paragraph about Enarx's purpose and its status as an open source project.

Enarx | Enarx

https://enarx.dev

Enarx

Search

# Enarx

Confidential Computing with WebAssembly

Download Try Enarx

100% Open Source

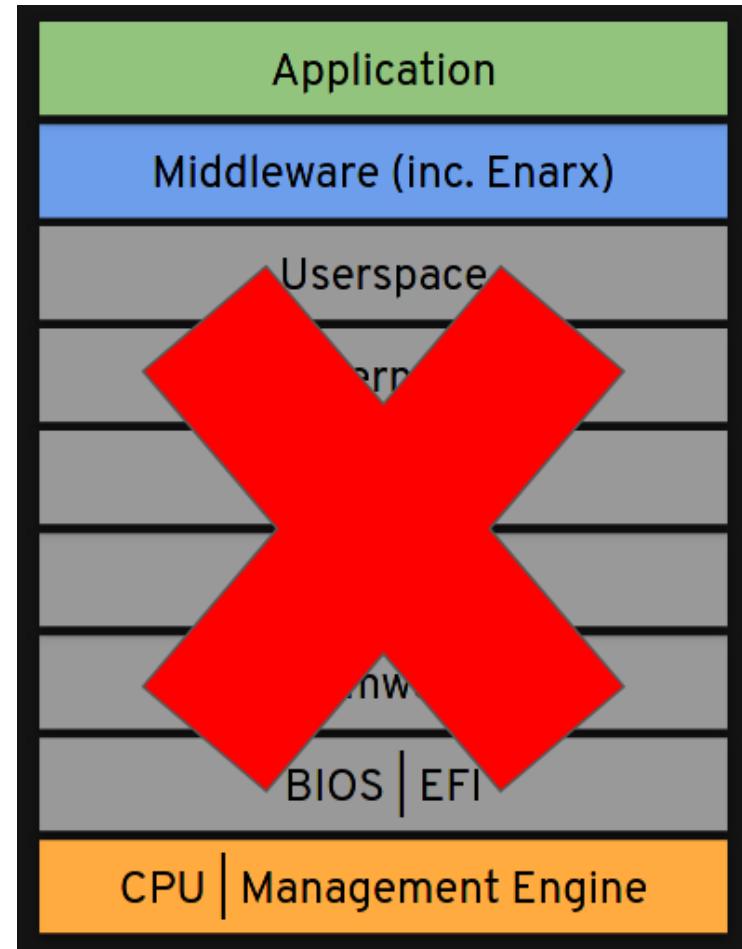
Enarx is the leading open source framework for running applications in TEEs (Trusted Execution Environments). It's part of the Confidential Computing Consortium from the Linux Foundation.



0101  
0101

# Enarx Threat Model

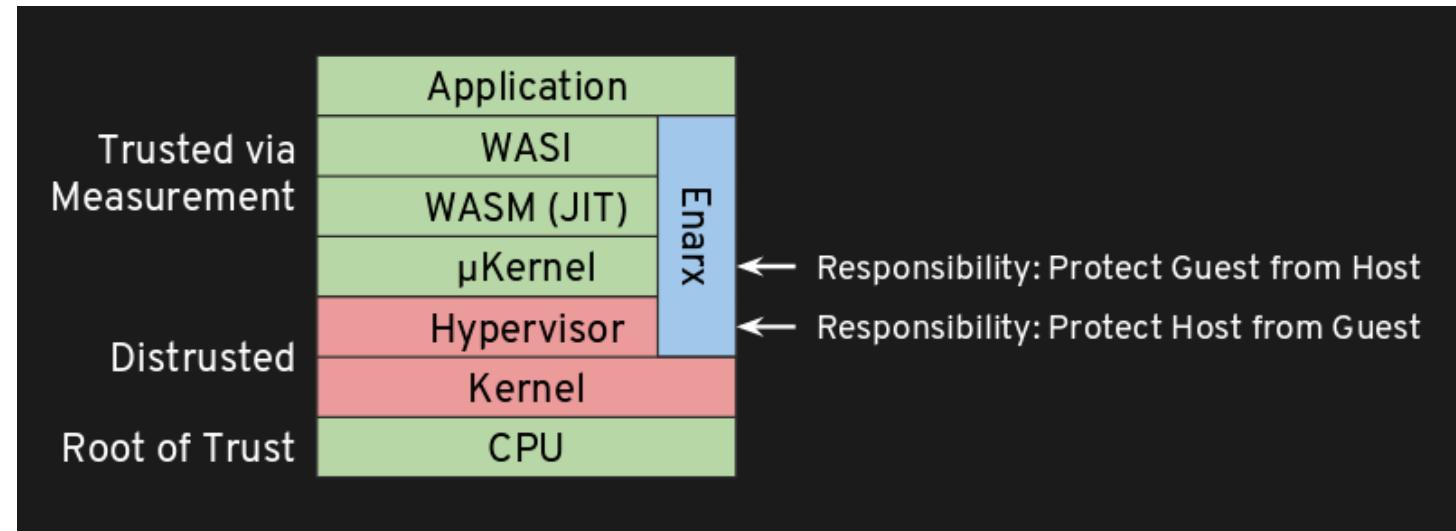
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





# Enarx

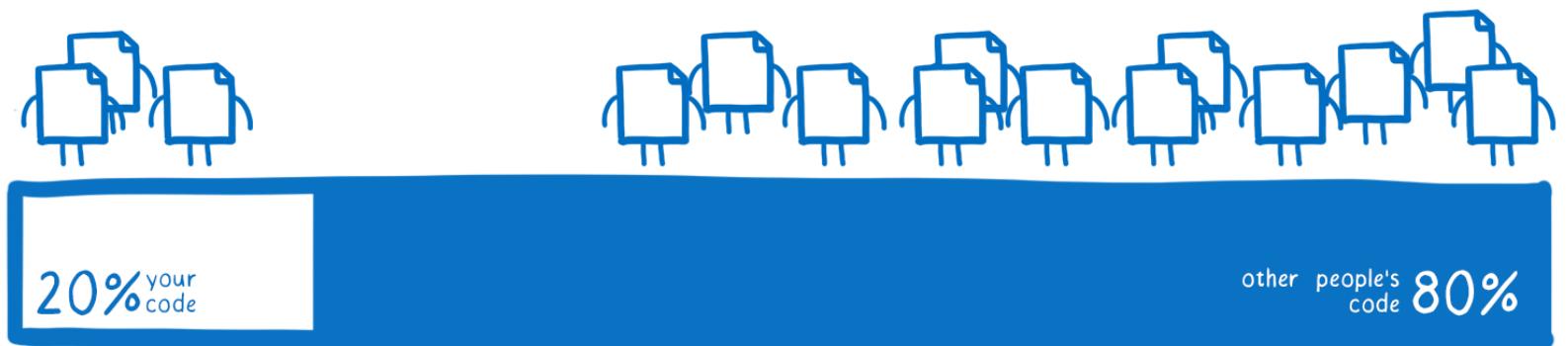
- Leverages Trusted Execution Environment (TEE) direct on processor
  - AMD's SEV, Intel's SGX and IBM's PEF
- Attestation of hardware and Enarx runtime



0101  
0101

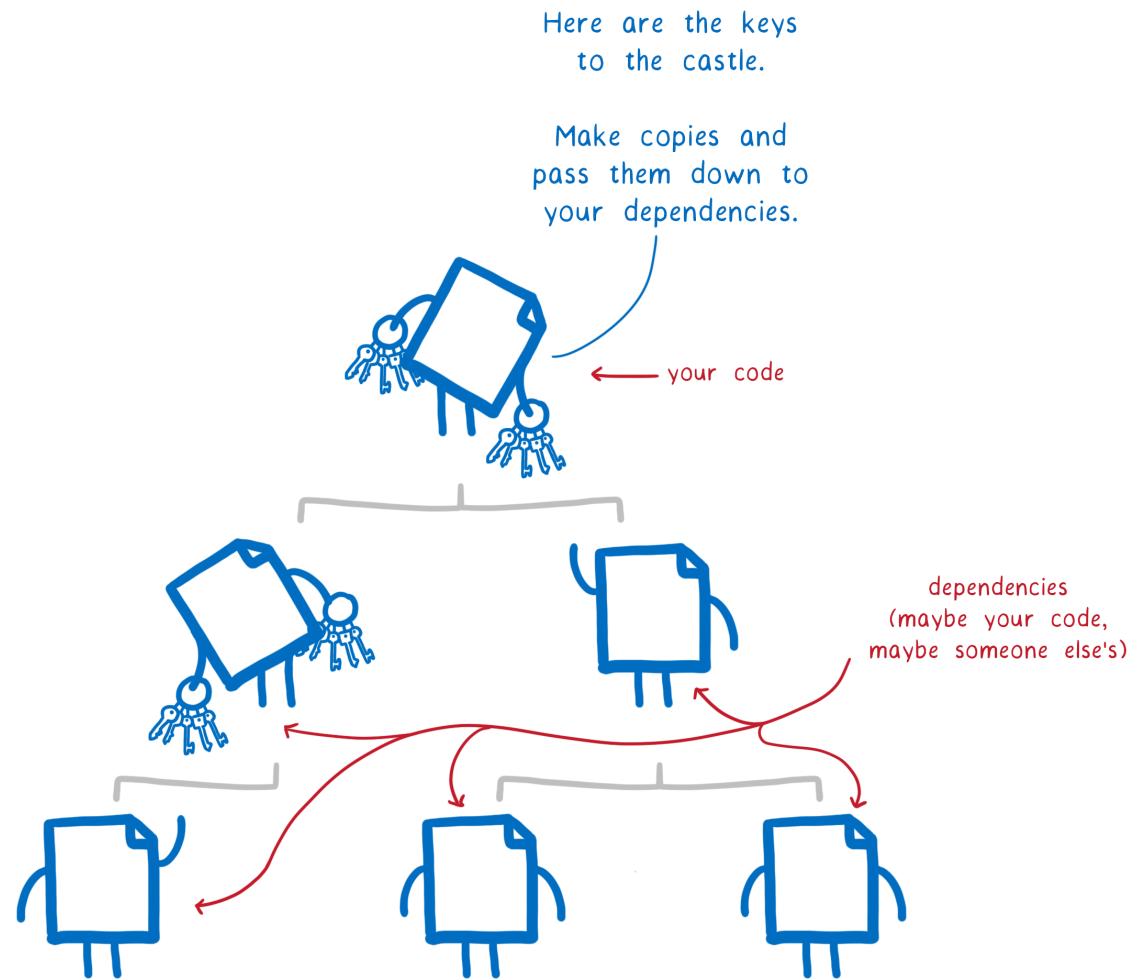
# WASM - What's next?

composition of an  
average code base



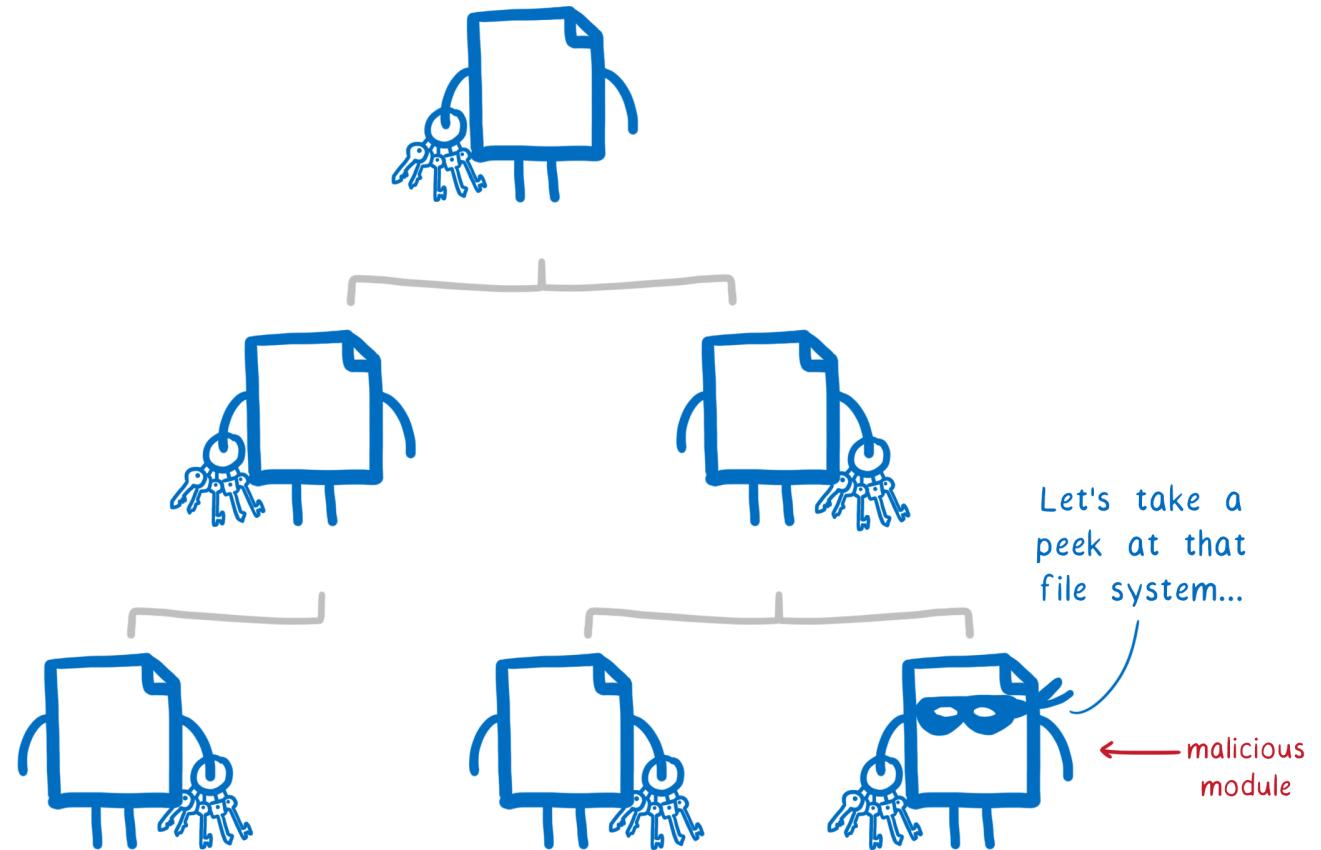
0101  
0101

# Dependencies



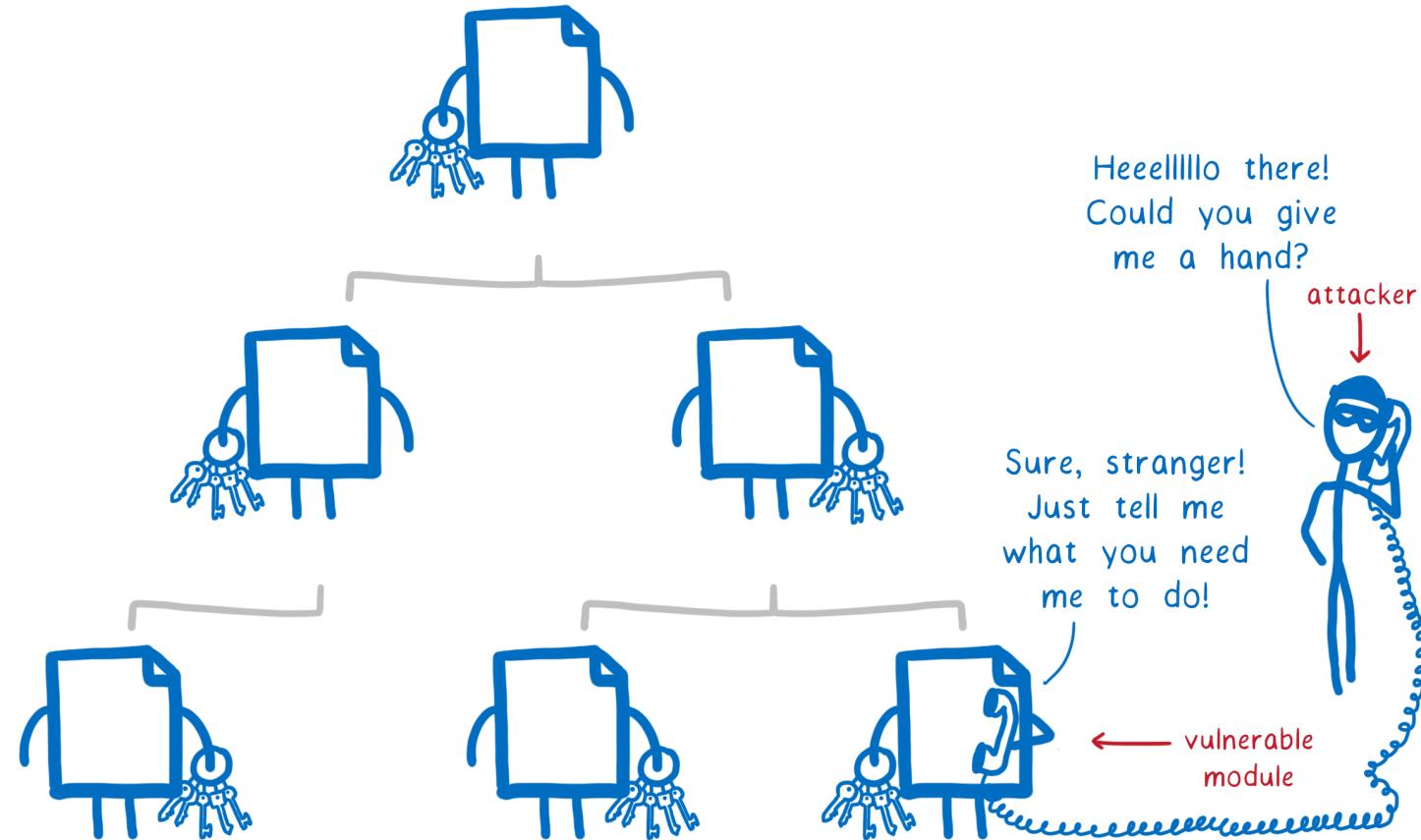
0101  
0101

# Malicious module



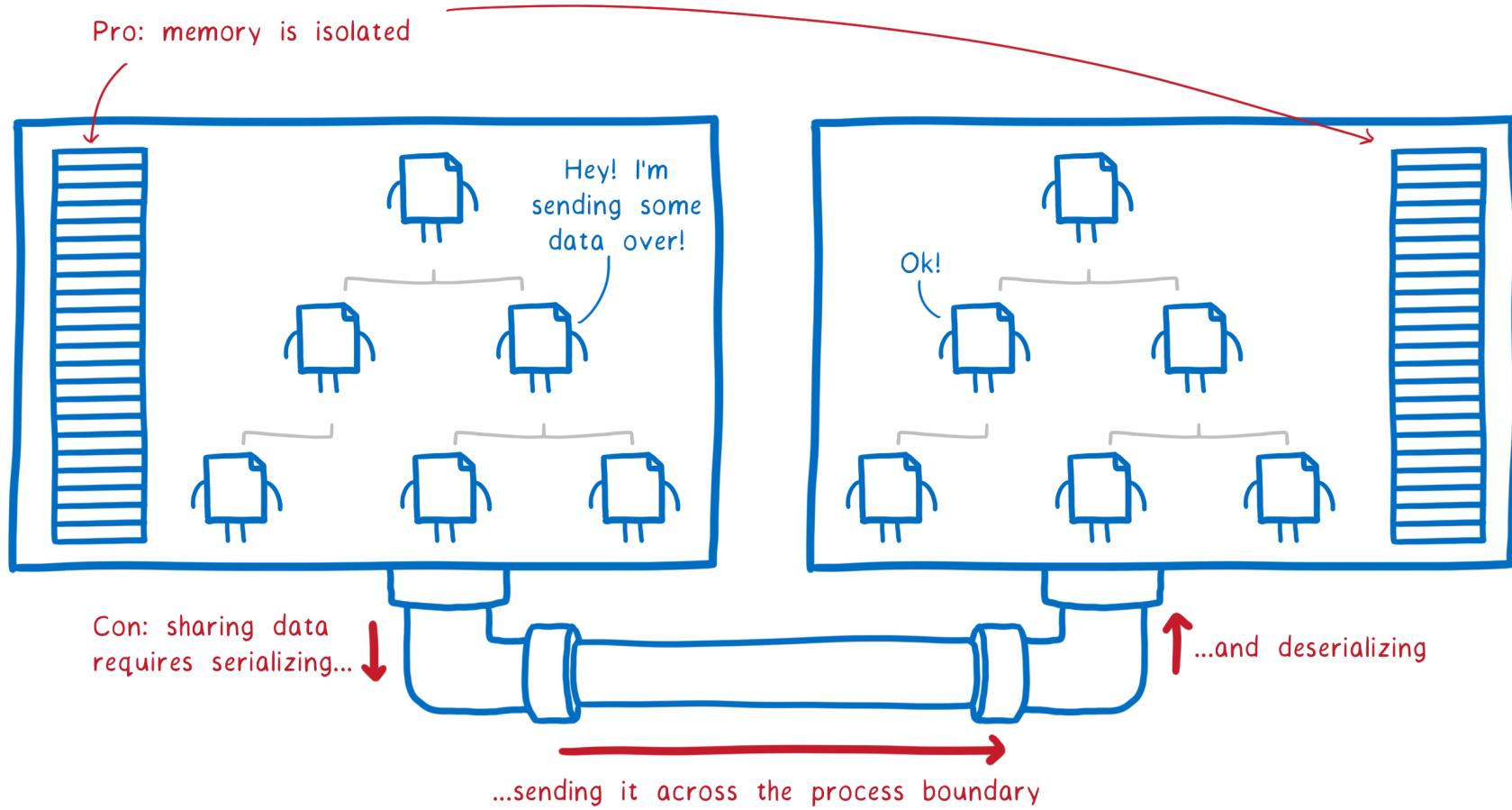
0101  
0101

# Vulnerable module



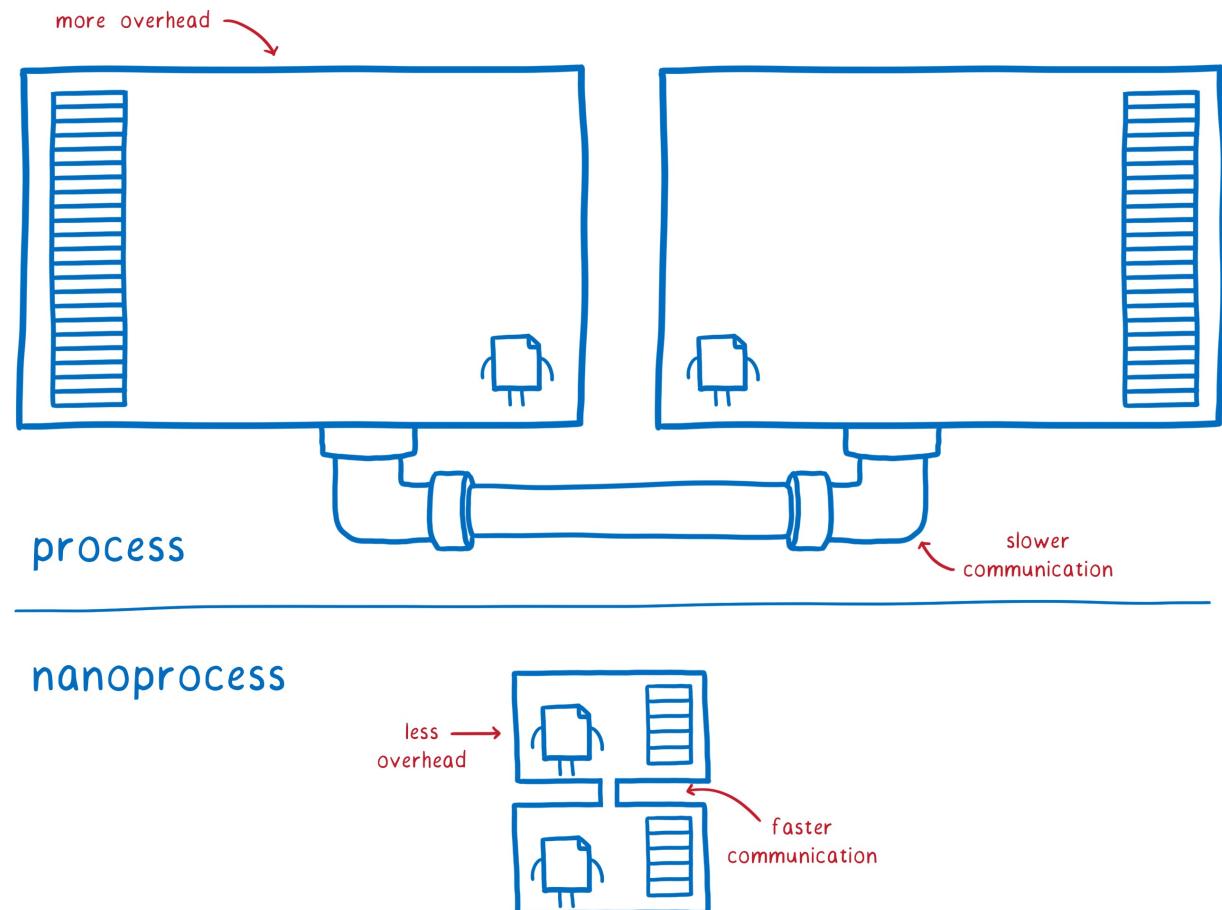
0101  
0101

# Process Isolation



0101  
0101

# WebAssembly Nano-Process



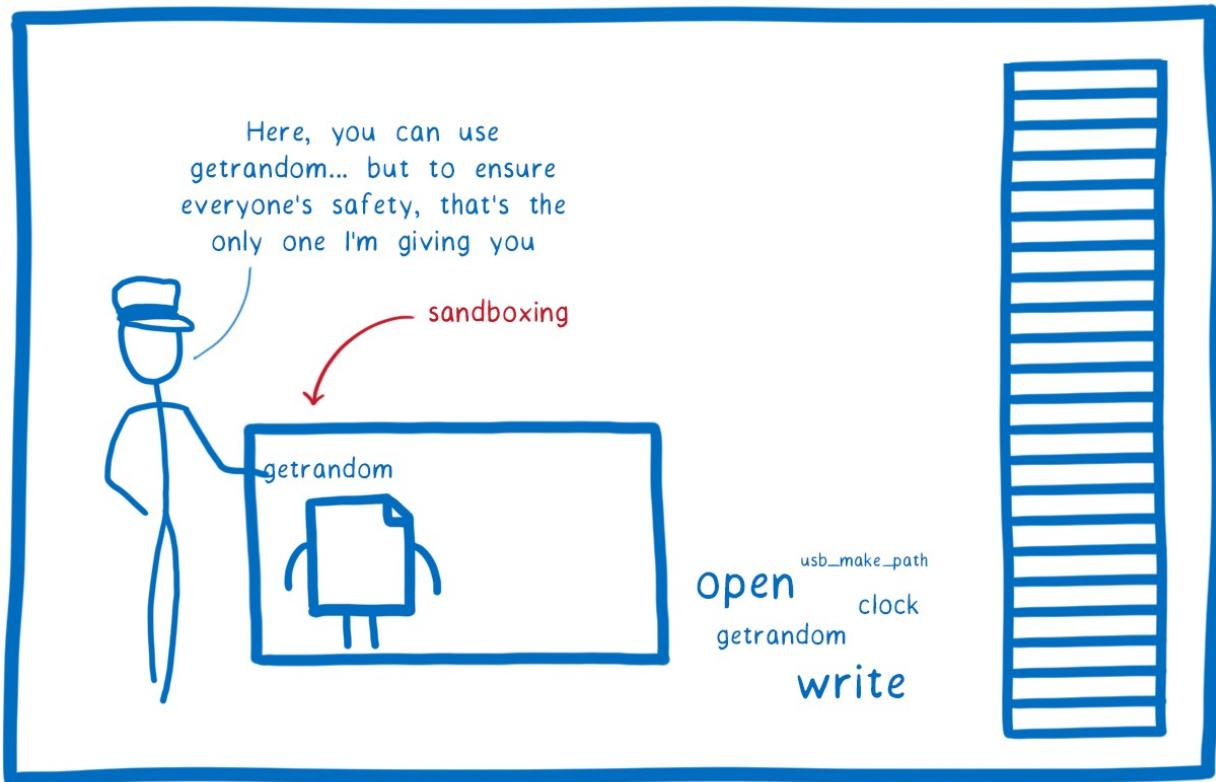
\* not drawn to scale



0101  
0101

# WebAssembly Nano-Process

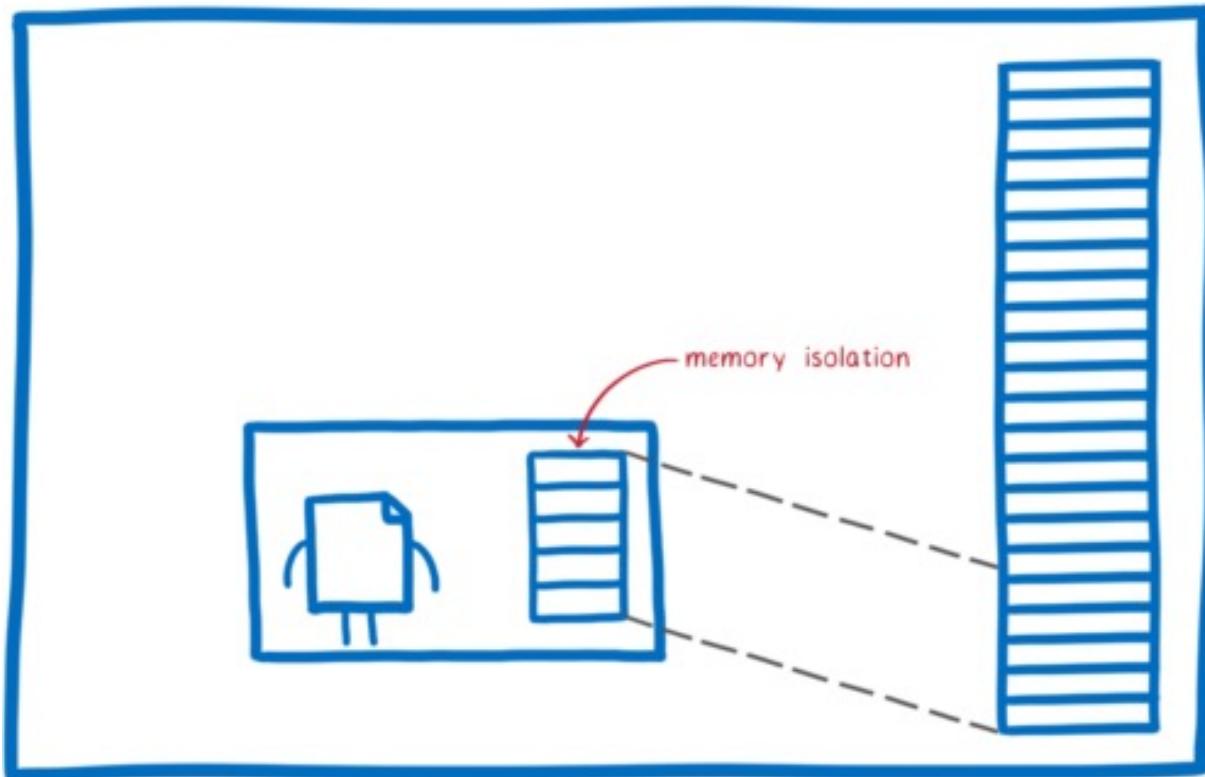
## 1. Sandboxing



0101  
0101

# WebAssembly Nano-Process

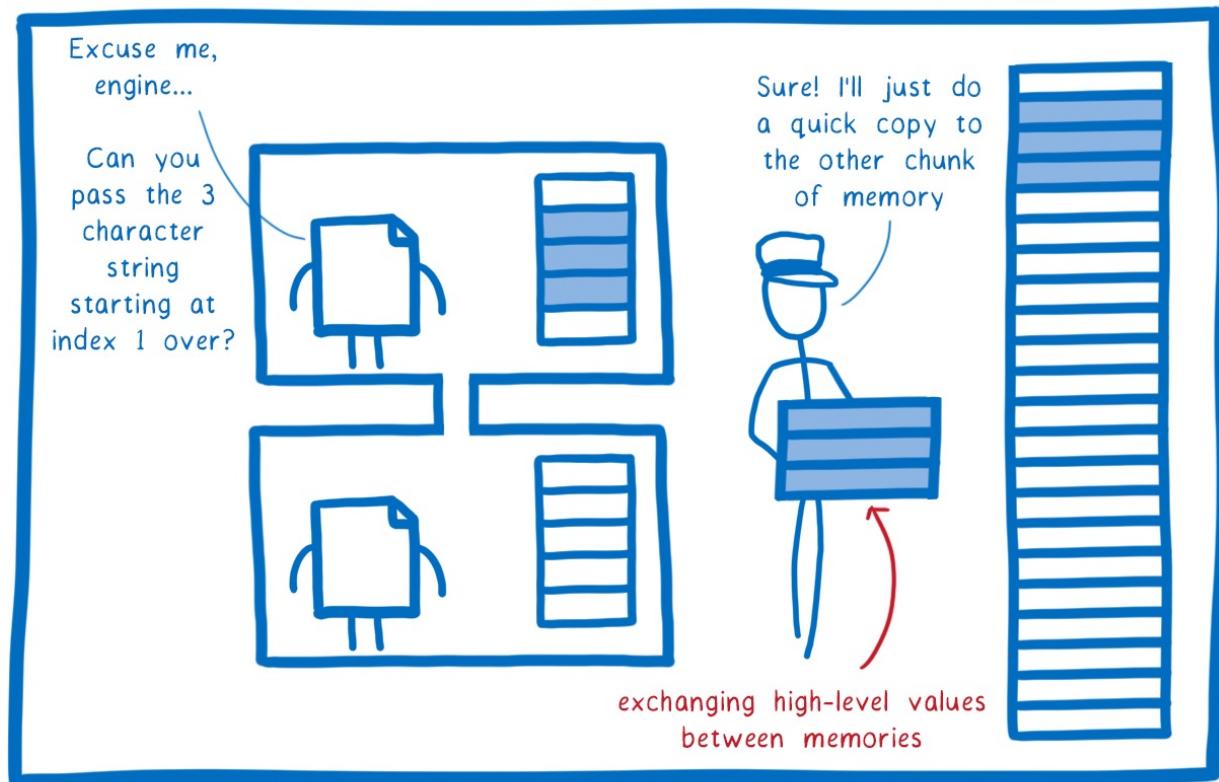
## 2. Memory model



0101  
0101

# WebAssembly Nano-Process

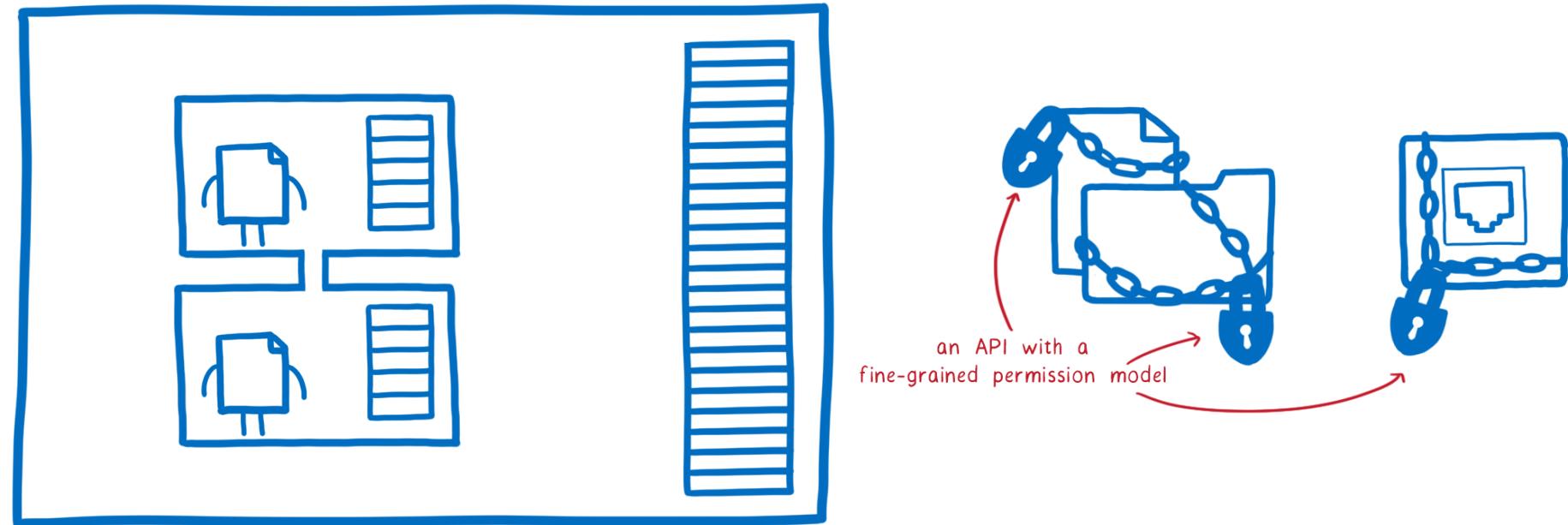
## 3. Interface Types



0101  
0101

# WebAssembly Nano-Process

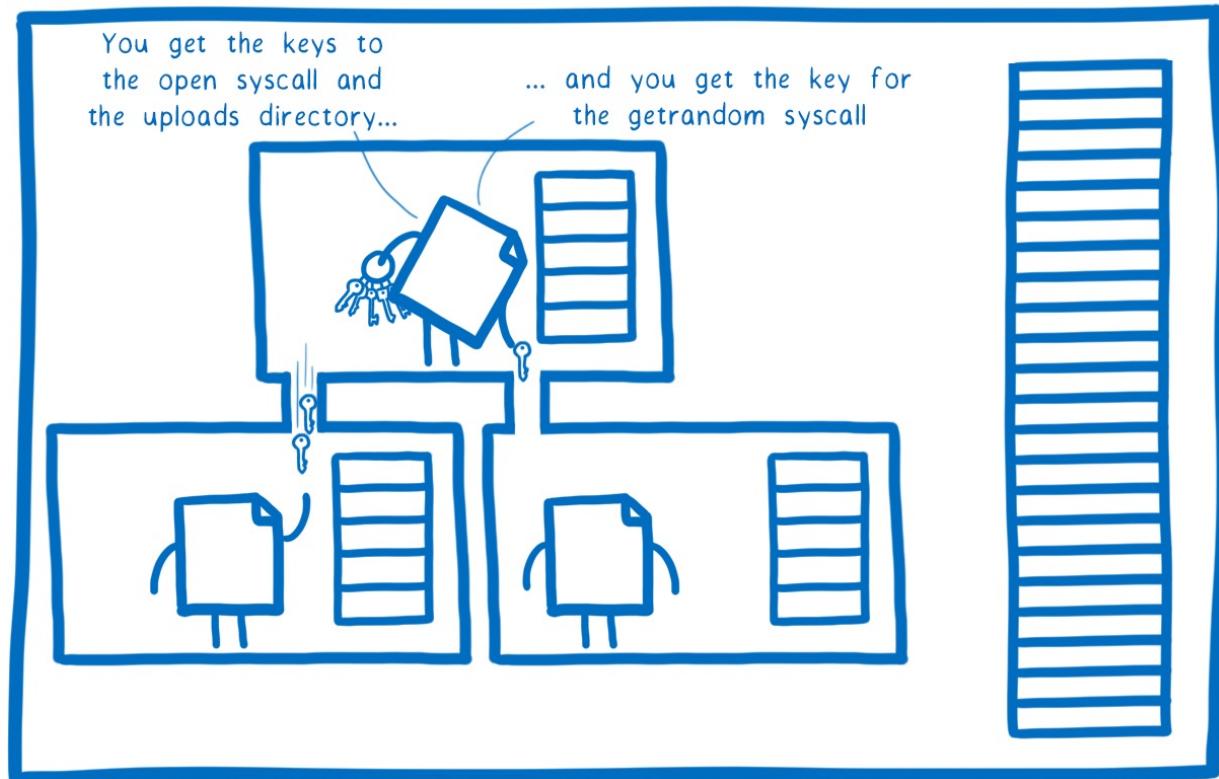
## 4. WebAssembly System Interface



0101  
0101

# WebAssembly Nano-Process

## 5. The missing link



0101  
0101

# WebAssembly Component Model

The screenshot shows a YouTube video player with the following details:

- Title:** Keynote: The Path to Components - Luke Wagner, Distinguished Engineer, Fastly
- Description:** There's a big new standards proposal we've been working on called the Component Model. It sits in-between WASI and Core WebAssembly and provides a way to compose code together like Lego bricks.
- Speaker:** Luke Wagner, Distinguished Engineer, Fastly
- Event:** Cloud Native Wasm Day NA 2022
- Duration:** 0:25 / 25:40
- Controls:** Standard YouTube controls for play, volume, and settings.





# Runtimes and Security

- Most security research published focusses on correctness of WASM runtimes/VM's
- Bytecode Alliance Blogpost September 2022:
  - “Security and Correctness in Wasmtime”
  - Written in Rust → Using all it's LangSec features
  - Continues Fuzzing & formal verification
  - Security process & vulnerability disclosure





# Conclusion

- Cloud Native ❤️ WebAssembly
- WebAssembly has a lot of potential to be used to run, extend, and secure your .NET applications
- Its as secure as the WebAssembly runtime implementation!
- I like top-down approach Bytecode Alliance is taking in moving forward, feedback will change



0101  
0101

# Questions?

- <https://github.com/nielstanis/updateconference2022>
- ntanis at Veracode.com
- <https://blog.fennec.dev>
- Děkuji! Thank you!

