

NDC { Porto }

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

Niels Tanis

VERACODE



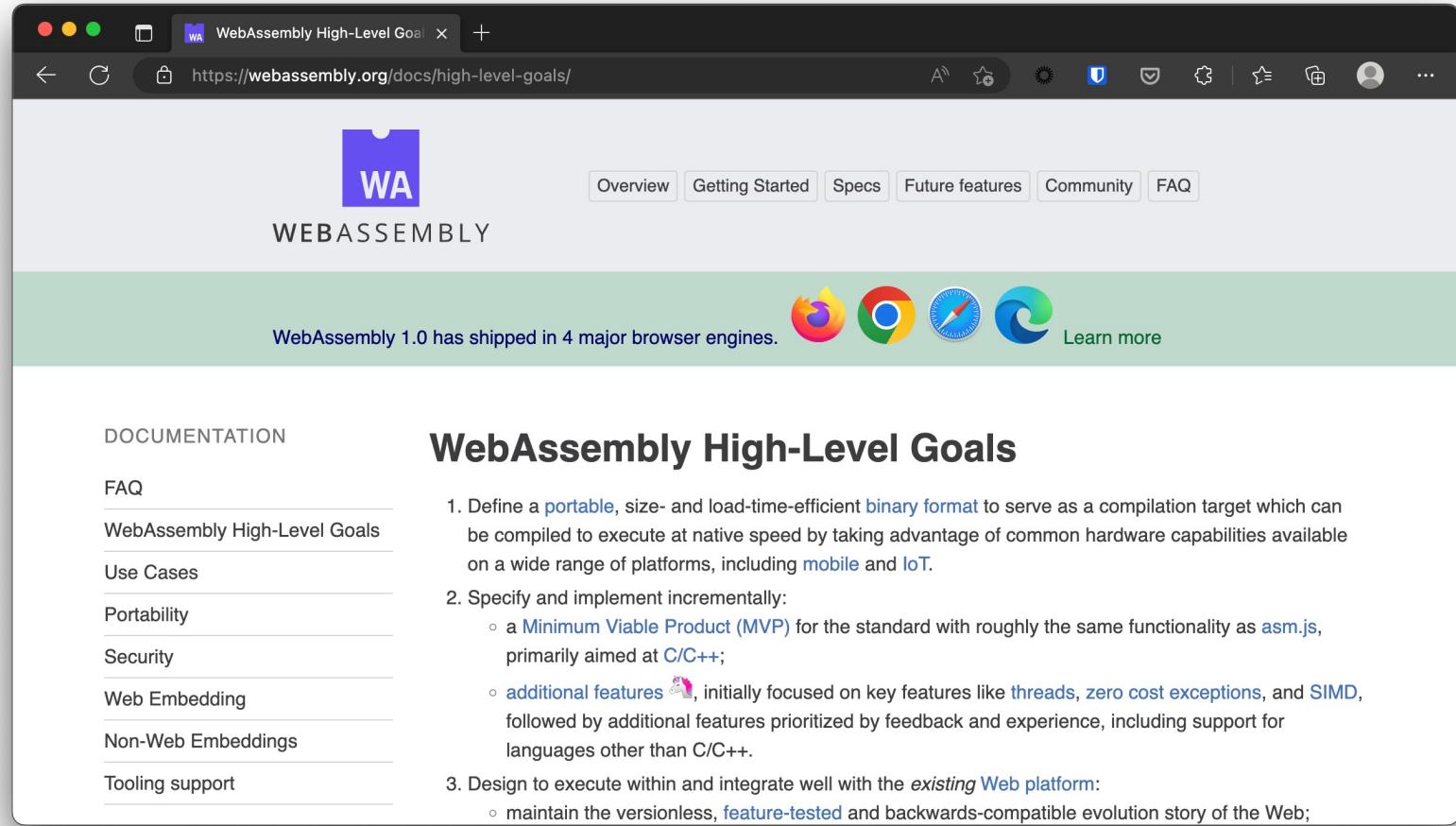
# Who am I?

- Niels Tanis
- Sr. Principal Security Researcher
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps
  - Enjoying Rust!
- Microsoft MVP - Developer Technologies



0101  
0101

# WebAssembly

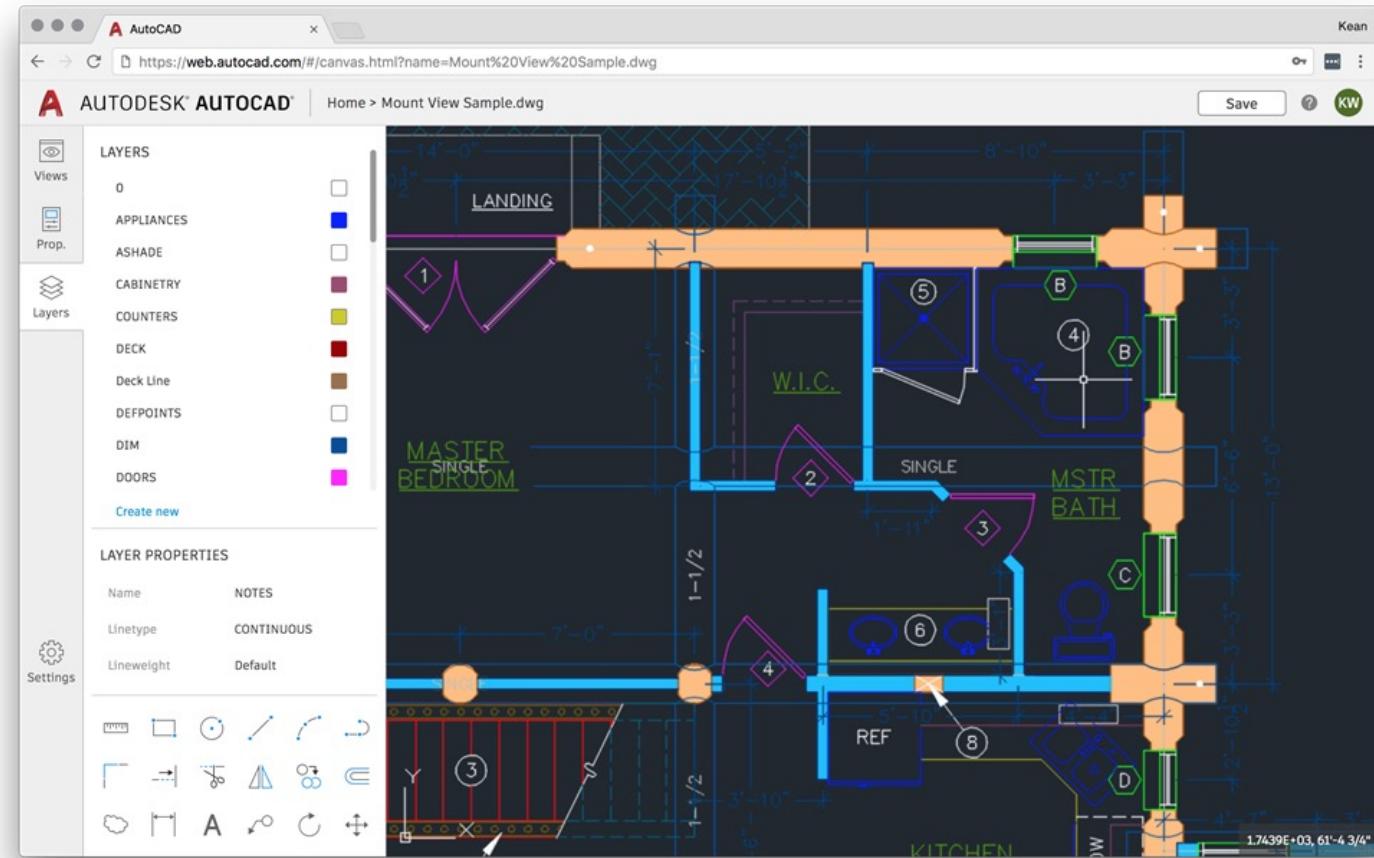


The screenshot shows a web browser window displaying the [WebAssembly High-Level Goals](https://webassembly.org/docs/high-level-goals/) page. The page has a dark header with a purple 'WA' logo and the word 'WEBASSEMBLY'. Below the header is a green banner stating 'WebAssembly 1.0 has shipped in 4 major browser engines.' followed by icons for Firefox, Chrome, Opera, and Edge, with a 'Learn more' link. The main content area is titled 'WebAssembly High-Level Goals' and lists three main goals:

- Define a portable, size- and load-time-efficient binary format to serve as a compilation target which can be compiled to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms, including mobile and IoT.
- Specify and implement incrementally:
  - a Minimum Viable Product (MVP) for the standard with roughly the same functionality as `asm.js`, primarily aimed at C/C++;
  - additional features 🎊, initially focused on key features like threads, zero cost exceptions, and SIMD, followed by additional features prioritized by feedback and experience, including support for languages other than C/C++.
- Design to execute within and integrate well with the existing Web platform:
  - maintain the versionless, feature-tested and backwards-compatible evolution story of the Web;



# WebAssembly - AutoCAD



# WebAssembly - SDK's



Published in [disney-streaming](#)

By [Mike Hanley](#) Follow  
Sep 8, 2021 · 10 min read · Listen

## Introducing the Disney+ Application Development Kit (ADK)

By: Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices

415 4

amazon | science

CLOUD AND SYSTEMS

## How Prime Video updates its app for more than 8,000 device types

The switch to WebAssembly increases stability, speed.

By [Alexandru Ene](#)  
January 27, 2022

Share

At [Prime Video](#), we're delivering content to millions of customers on



# Agenda

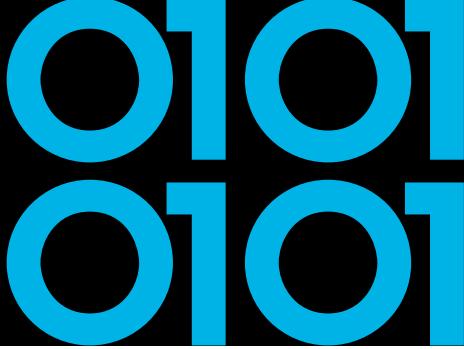
- Introduction
- WebAssembly 101
- Running .NET on WebAssembly
- Extending .NET with WebAssembly
- Securing .NET with WebAssembly
- Conclusion
- Q&A



# 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





# WebAssembly

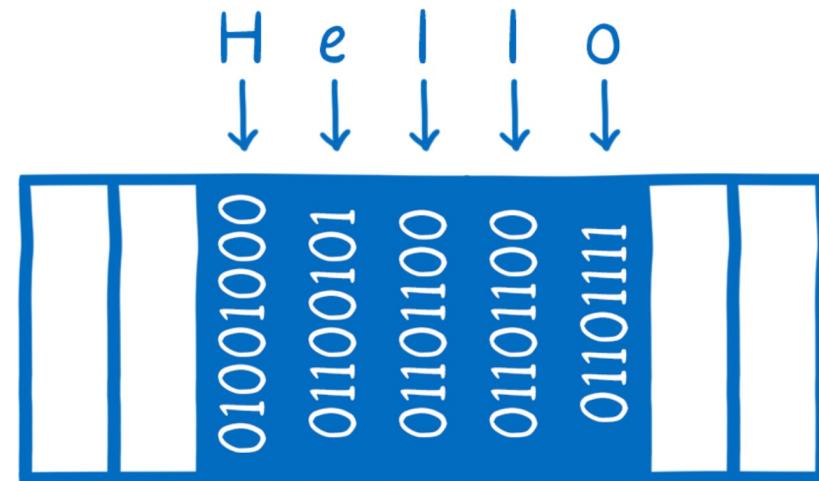
- Binary instruction format for stack-based virtual machine similar to .NET running MSIL
- Designed as a portable compilation target
- The security model of WebAssembly:
  - Protect users from buggy or malicious modules
  - Provide developers with useful primitives and mitigations for developing safe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



0101  
0101

# WebAssembly Memory

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





# WebAssembly Control-Flow Integrity

```
int number = Convert.ToInt32(Console.ReadLine());
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!");
```



# WebAssembly Control-Flow Integrity

```
int number = Convert.ToInt32(Console.ReadLine());  
Console.WriteLine($"Number {number}");  
if (number>5)
```

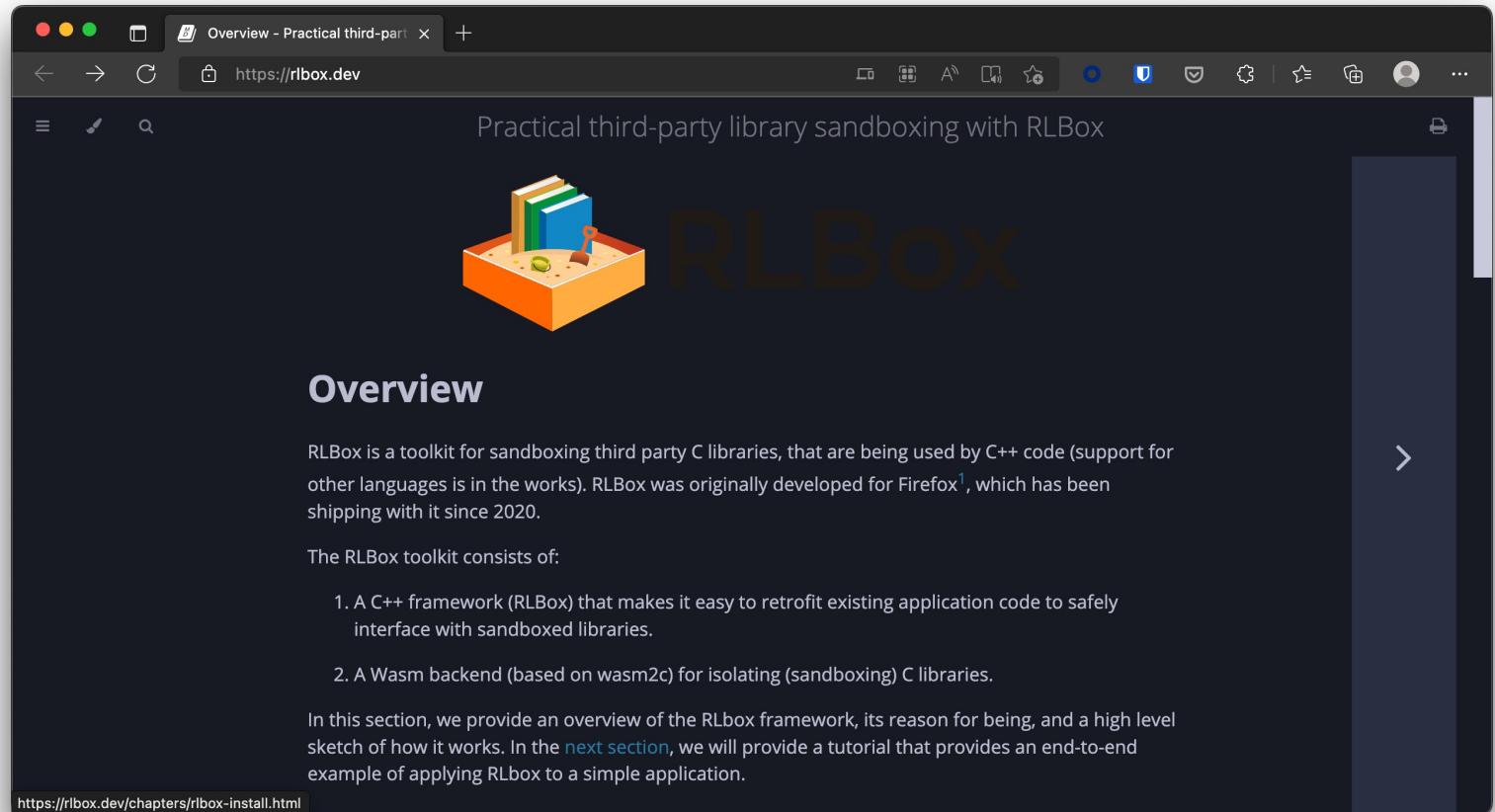
```
Console.WriteLine("Number is larger than 5");
```

```
Console.WriteLine("Number is smaller than 5");
```

```
Console.WriteLine("Done!");
```

0101  
0101

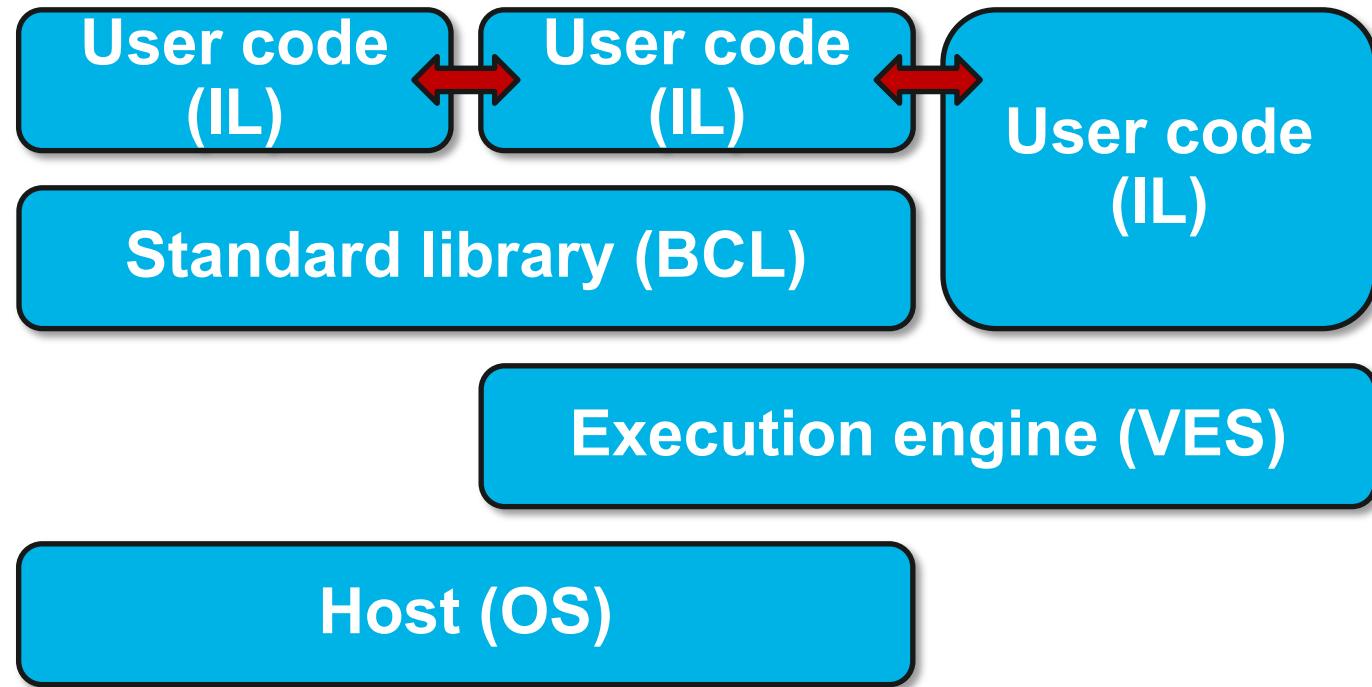
# FireFox RLBox



The screenshot shows a Firefox browser window with a dark theme. The title bar says "Overview - Practical third-part" and the address bar shows "https://rlbox.dev". The main content area has a dark background with a central logo featuring three books in a sandcastle-like setup with a shovel. To the right of the logo is the word "RLBox" in large, bold, white letters. Below the logo, the word "Overview" is centered in a large, bold, white font. The text below "Overview" describes RLBox as a toolkit for sandboxing third-party C libraries. It mentions support for C++ and other languages, its development for Firefox since 2020, and its components: a C++ framework and a Wasm backend. A link at the bottom left points to "https://rlbox.dev/chapters/rlbox-install.html".

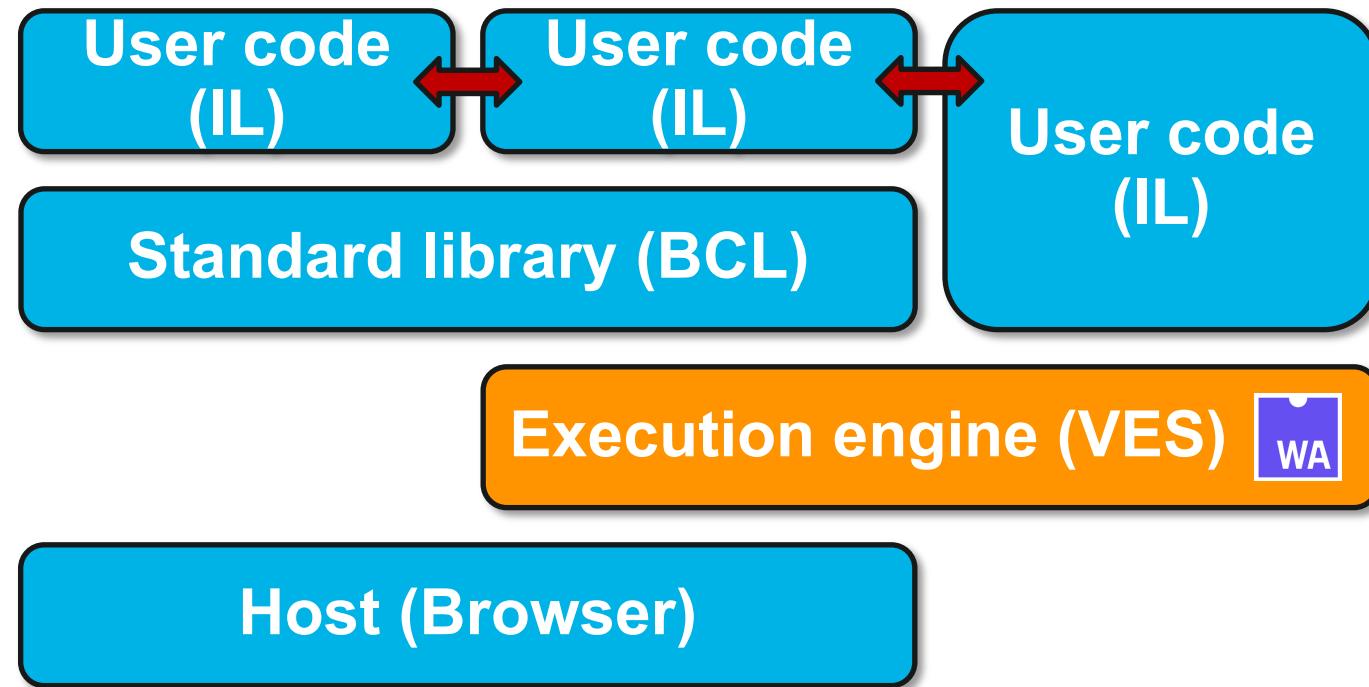


# Running .NET on WebAssembly



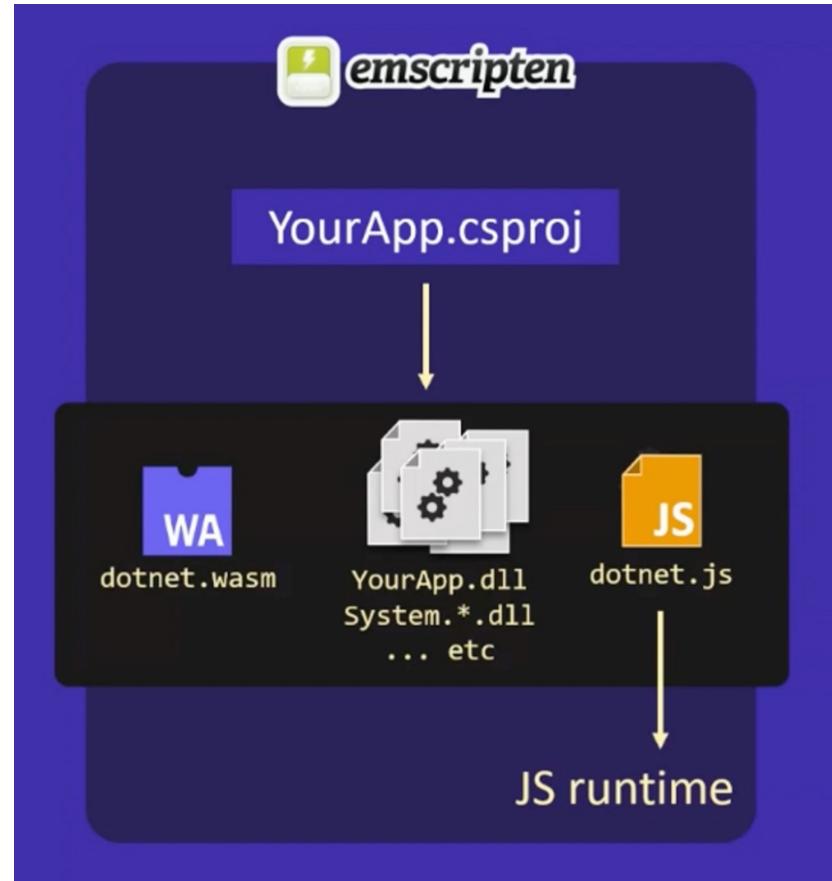


# Running .NET on WebAssembly



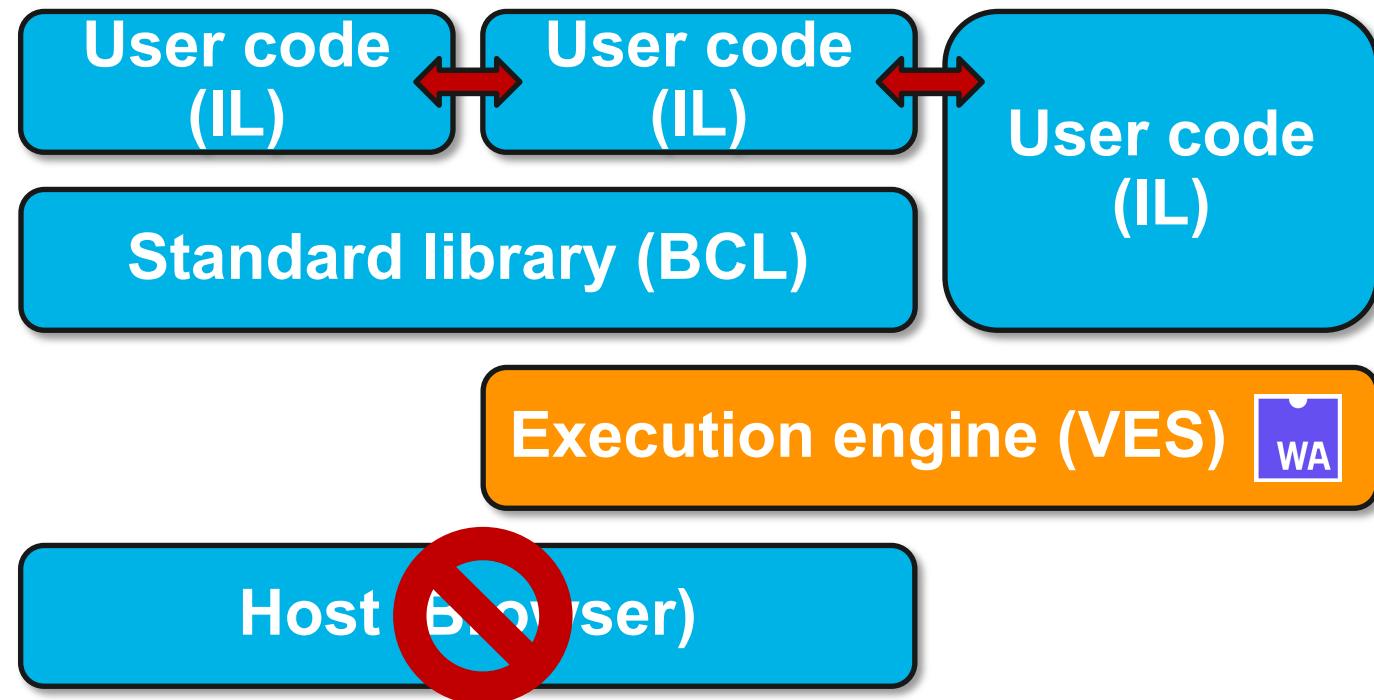


# Blazor WebAssembly





# 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

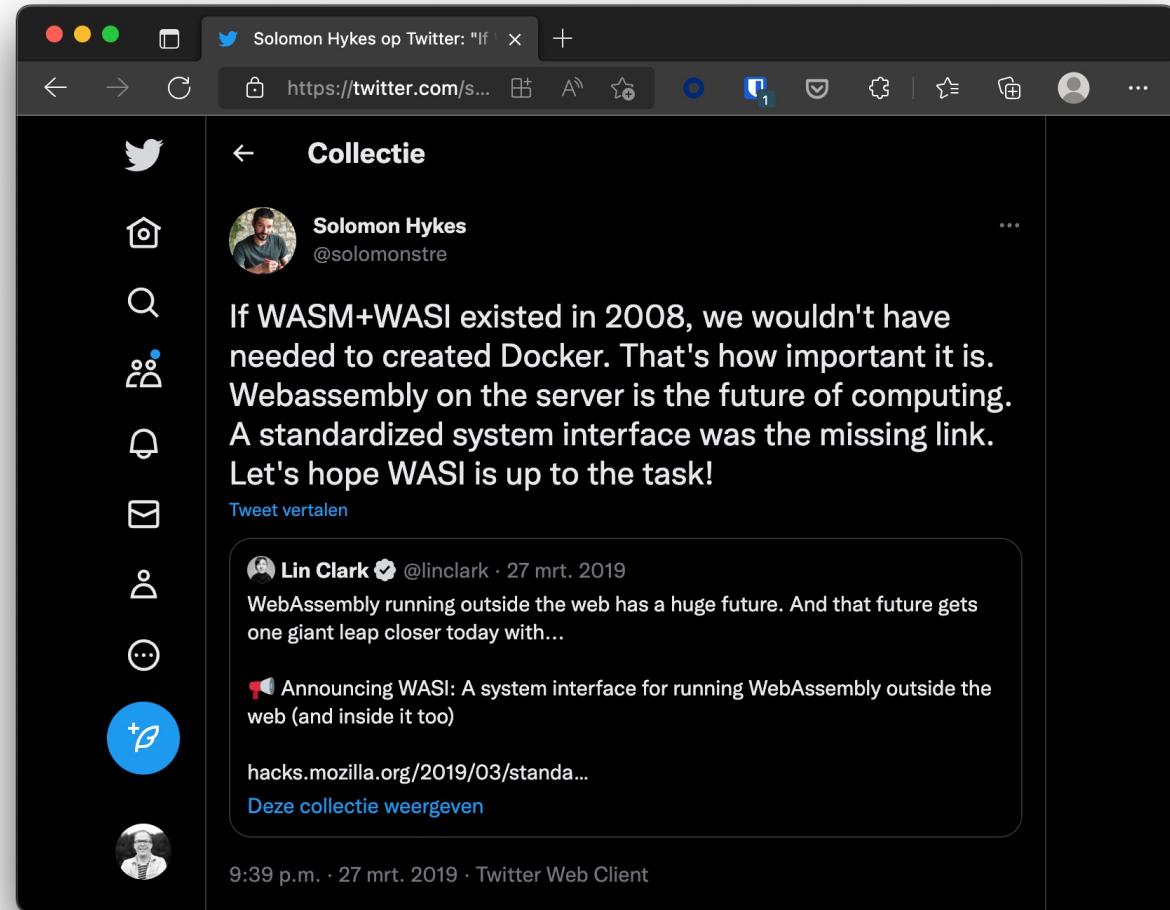
# WebAssembly System Interface WASI



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

0101  
0101

# Docker vs WASM & WASI



0101  
0101

# Docker vs WASM & WASI

A screenshot of a Twitter web application window. The URL in the address bar is <https://twitter.com/s...>. 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 :). Below the main tweet is a reply from Solomon Hykes (@solomonstre) dated 27 mrt. 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/status/110341144000000000](https://twitter.com/linclark/status/110341144000000000)". The bottom of the tweet shows engagement statistics: 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

# Docker & WASM

0101  
0101

The screenshot shows a web browser window with the title "Introducing the Docker+Wasm Technical Preview". The page features a purple header bar with the text "Wasm is a fast, light alternative to Linux containers — try it out today in the Docker+Wasm Technical Preview". Below this is the Docker+Wasm logo. The main content area has a large heading "Introducing the Docker+Wasm Technical Preview". It includes a photo of Michael Irwin, the author, and the date "Oct 24 2022". A paragraph at the bottom states: "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."

The screenshot shows a diagram of the Docker Engine architecture. At the top is the "Docker Engine", which connects to a central "containerd" component. The "containerd" component manages three separate container instances. Each container instance consists of a "containerd-shim" layer, a "runc" layer, and a "Container process". The third container instance also includes a "wasmedge" layer and a "Wasm Module". Arrows indicate the flow of communication between these layers.

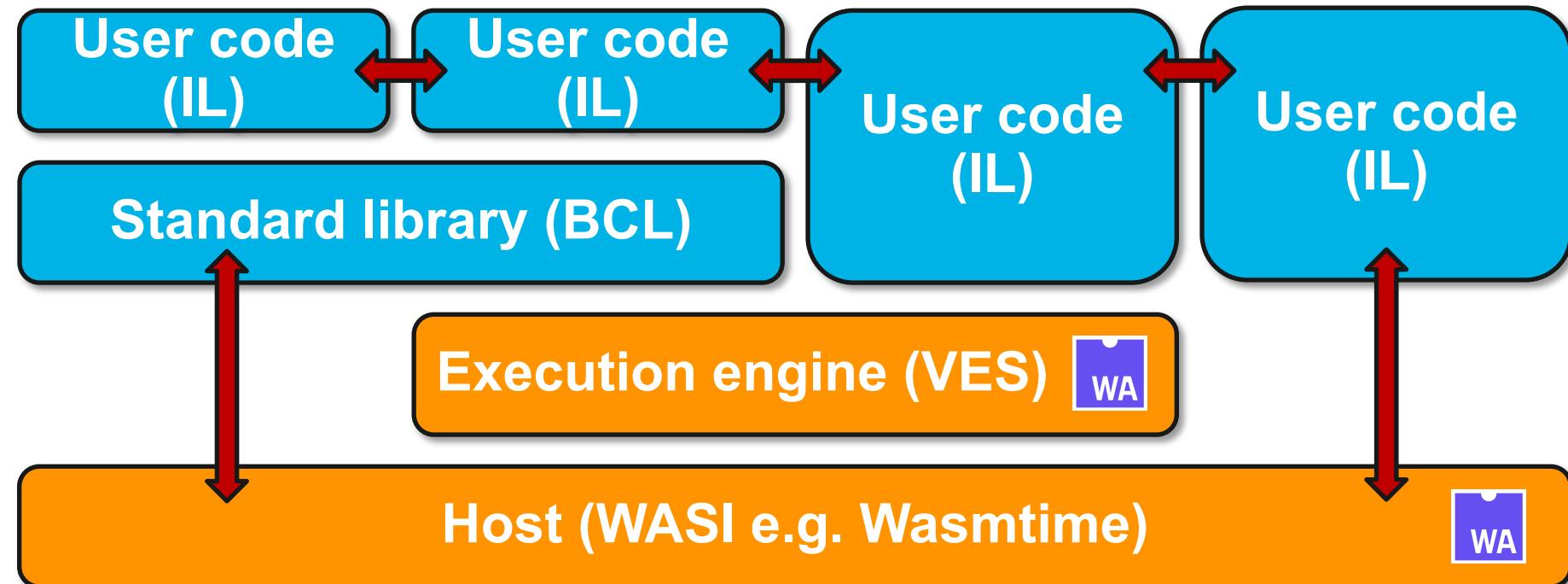
**Let's look at an example!**

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



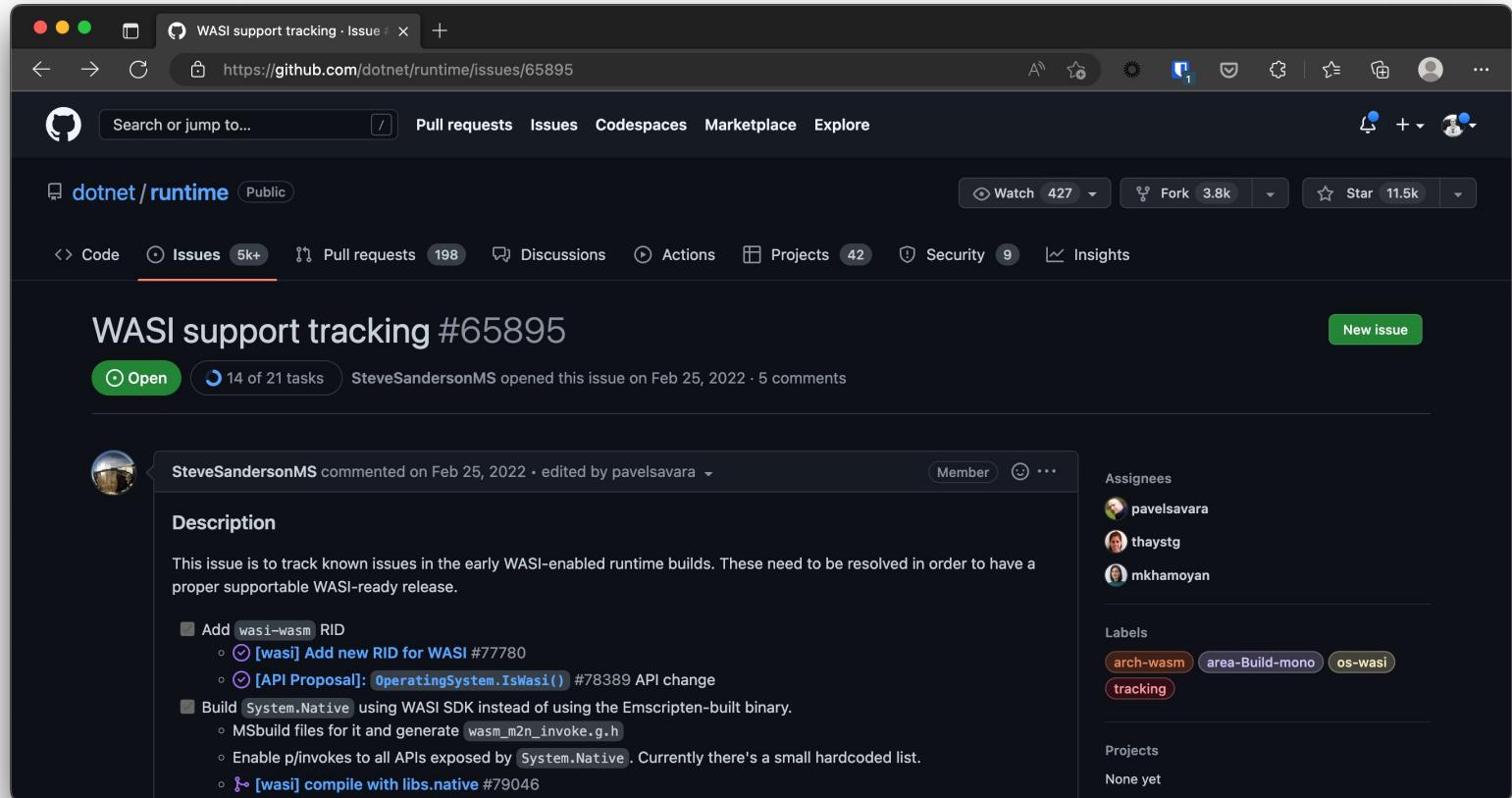


# Experimental WASI SDK for .NET



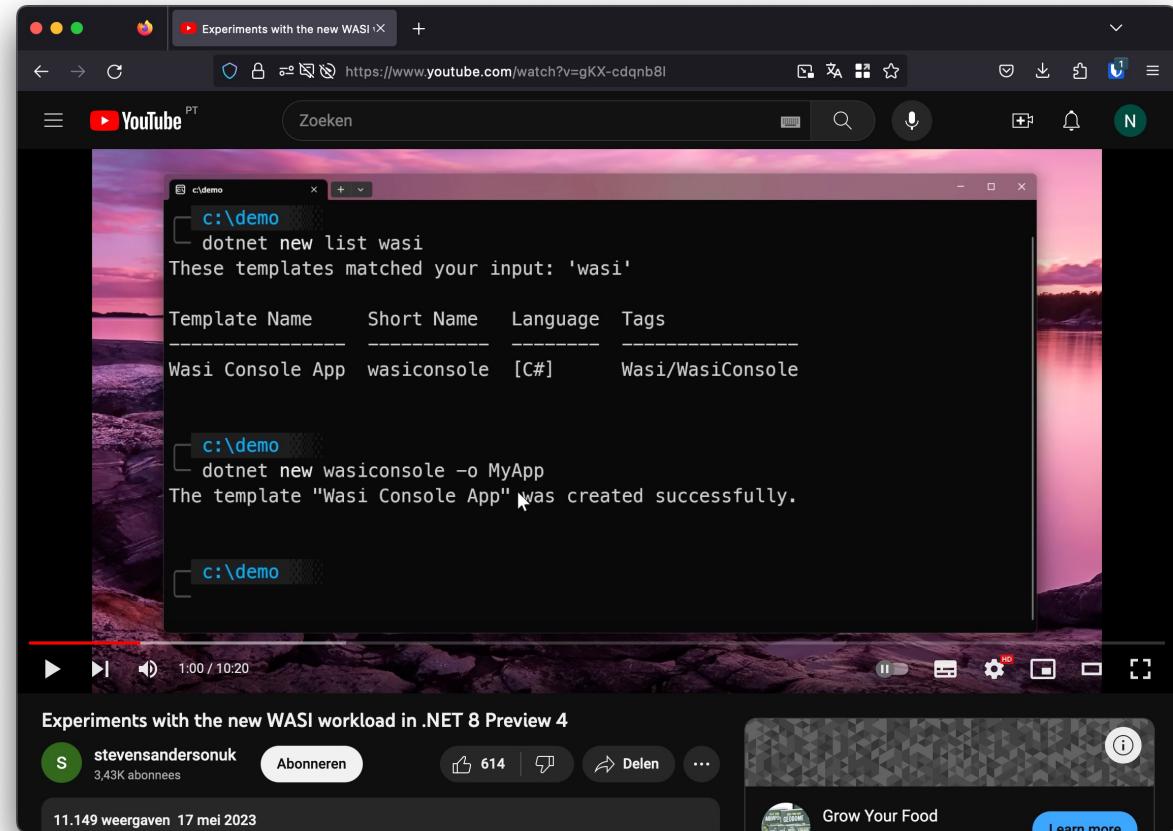
# Experimental WASI SDK for .NET

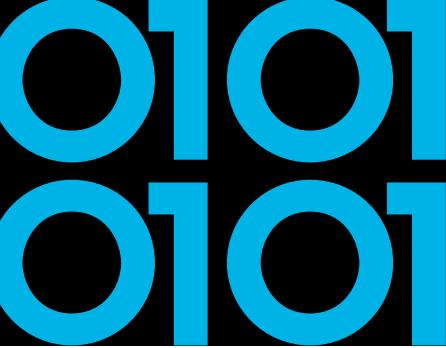
0101  
0101



0101  
0101

# Experimental WASI workload .NET8



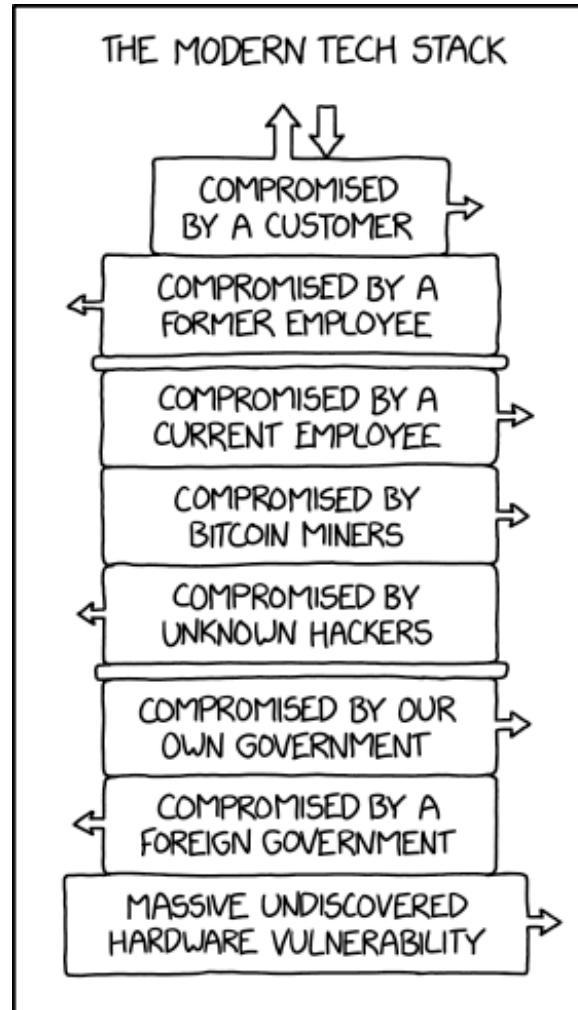


# Extending .NET with WASM

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

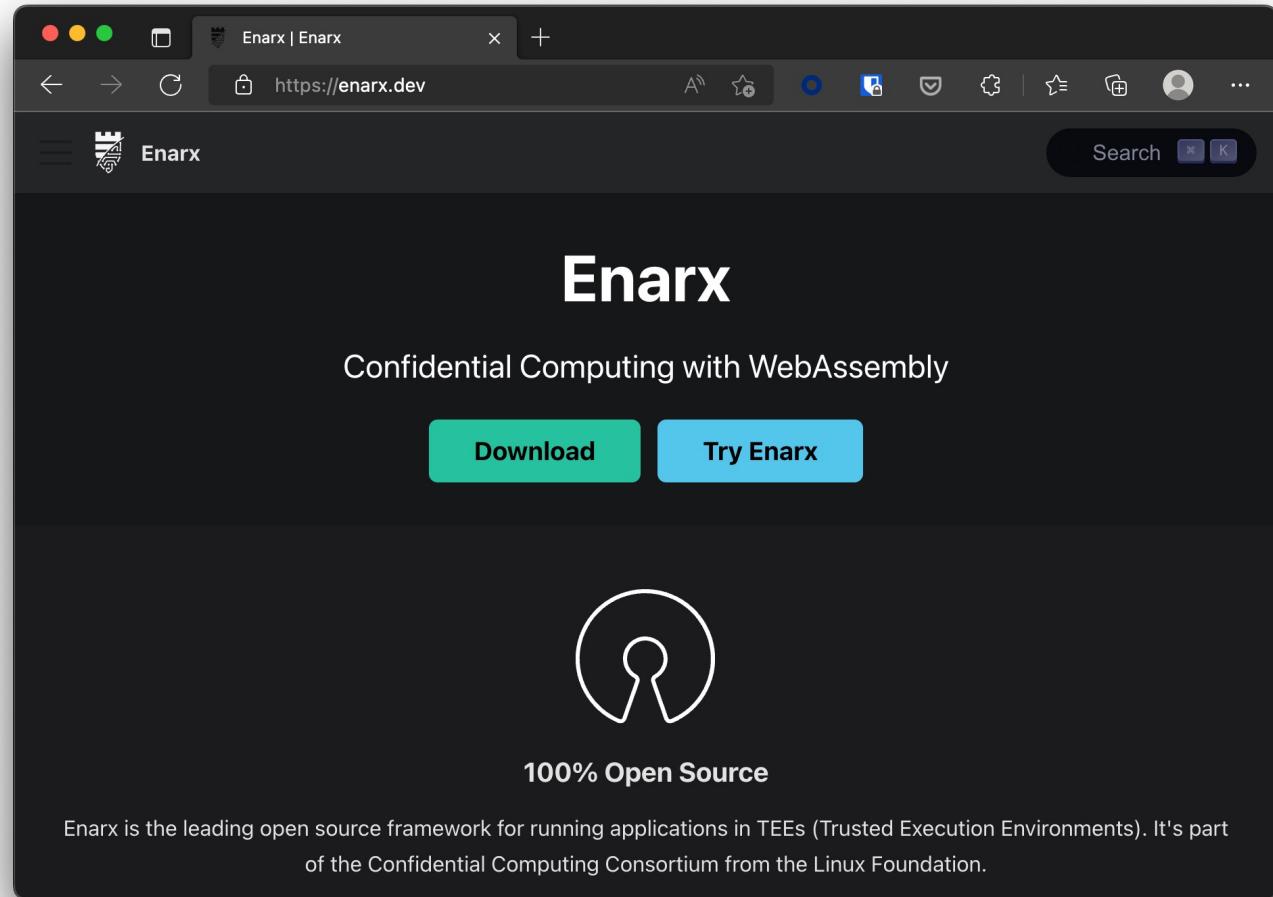
0101  
0101

# Trusted Computing - XKCD 2166



0101  
0101

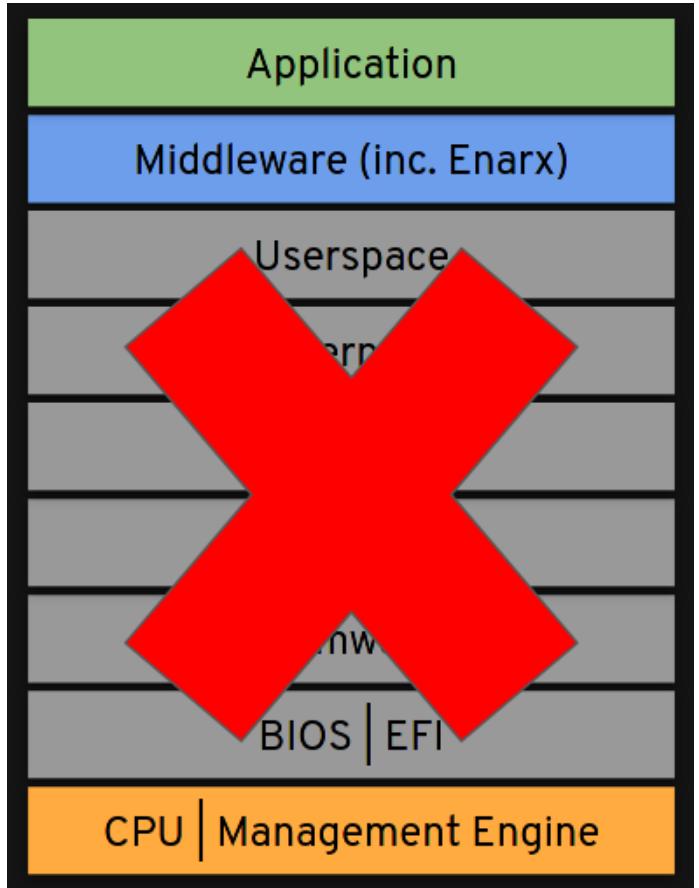
# Enarx



# Enarx Threat Model

0101  
0101

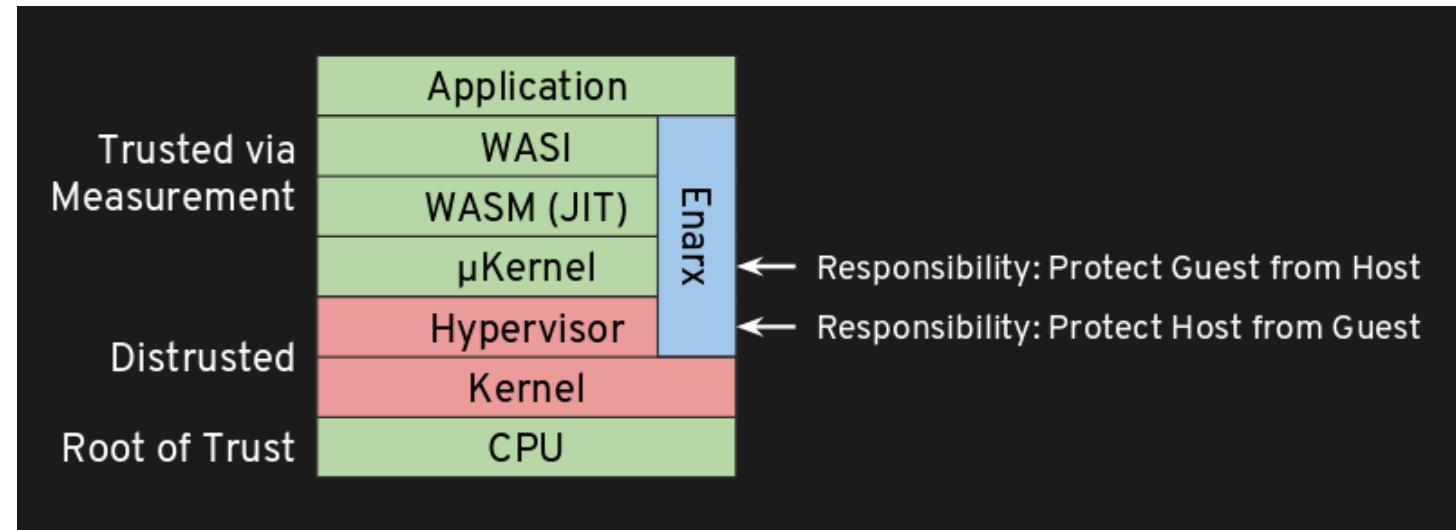
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





# 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



# Confidential Computing Consortium

0101  
0101

The screenshot shows a web browser window displaying the Confidential Computing Consortium's website. The URL in the address bar is <https://confidentialcomputing.io/projects/current-projects/>. The page header includes the Linux Foundation logo and the Confidential Computing Consortium logo. The main content area features a red circular graphic on the right. Text on the page includes: "A community focused on open source licensed projects securing data in use & accelerating the adoption of confidential computing through open collaboration.", "Every member is welcome; every project meeting our criteria is welcome.", and "We are a transparent, collaborative community.". Below this, there is a section for the Enarx project, featuring its logo, name, a brief description, and a "LEARN MORE" button.

THE LINUX FOUNDATION PROJECTS

CONFIDENTIAL COMPUTING CONSORTIUM

About Projects Resources Get Involved BECOME A MEMBER

A community focused on open source licensed projects securing data in use & accelerating the adoption of confidential computing through open collaboration.

Every member is welcome; every project meeting our criteria is welcome.

We are a transparent, collaborative community.

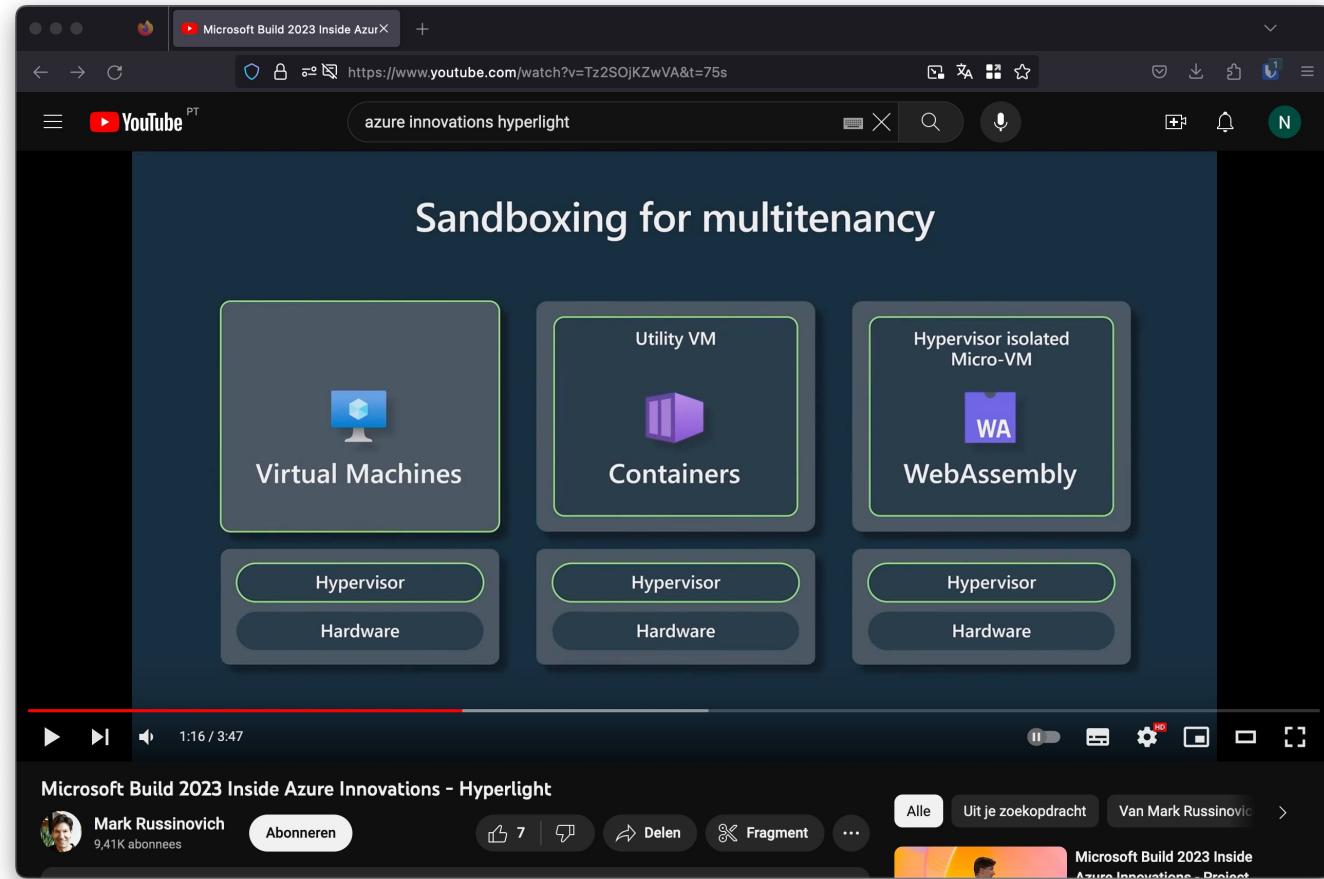
**Enarx**

Enarx provides a platform abstraction for Trusted Execution Environments (TEEs) enabling creating and running “private, fungible, serverless” applications.

LEARN MORE

0101  
0101

# Project Hyperlight



# DotNetIsolator & Project Hyperlight



DotNetIsolator: an experimental package for running .NET code in an isolated sandbox

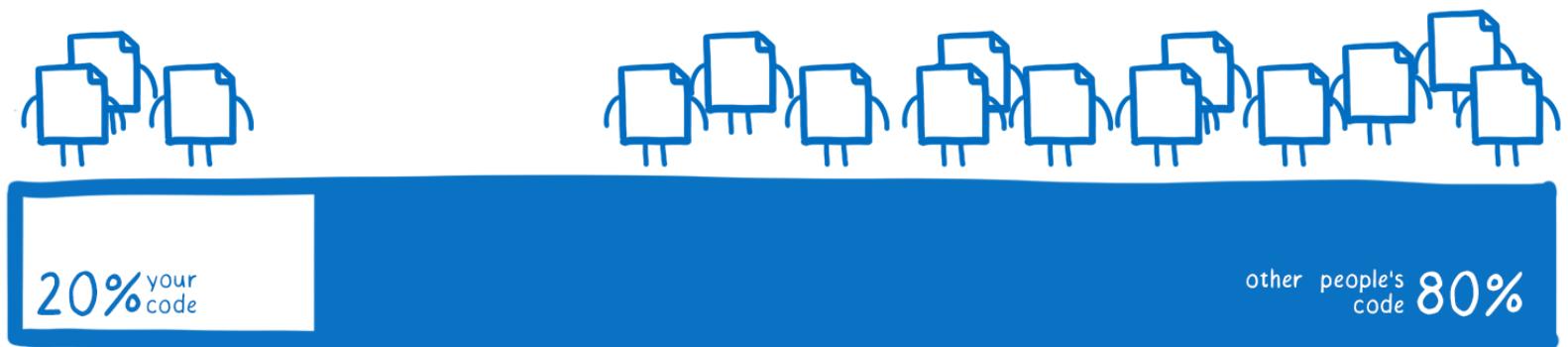
Sandboxing Your Sandbox: Leveraging Hypervisors for WebAssembly Security  
By: Danilo (Dan) Chiarlone

Sandboxing Your Sandbox: Leveraging Hypervisors for WebAssembly Security - Danilo (Dan) Chiarlone

0101  
0101

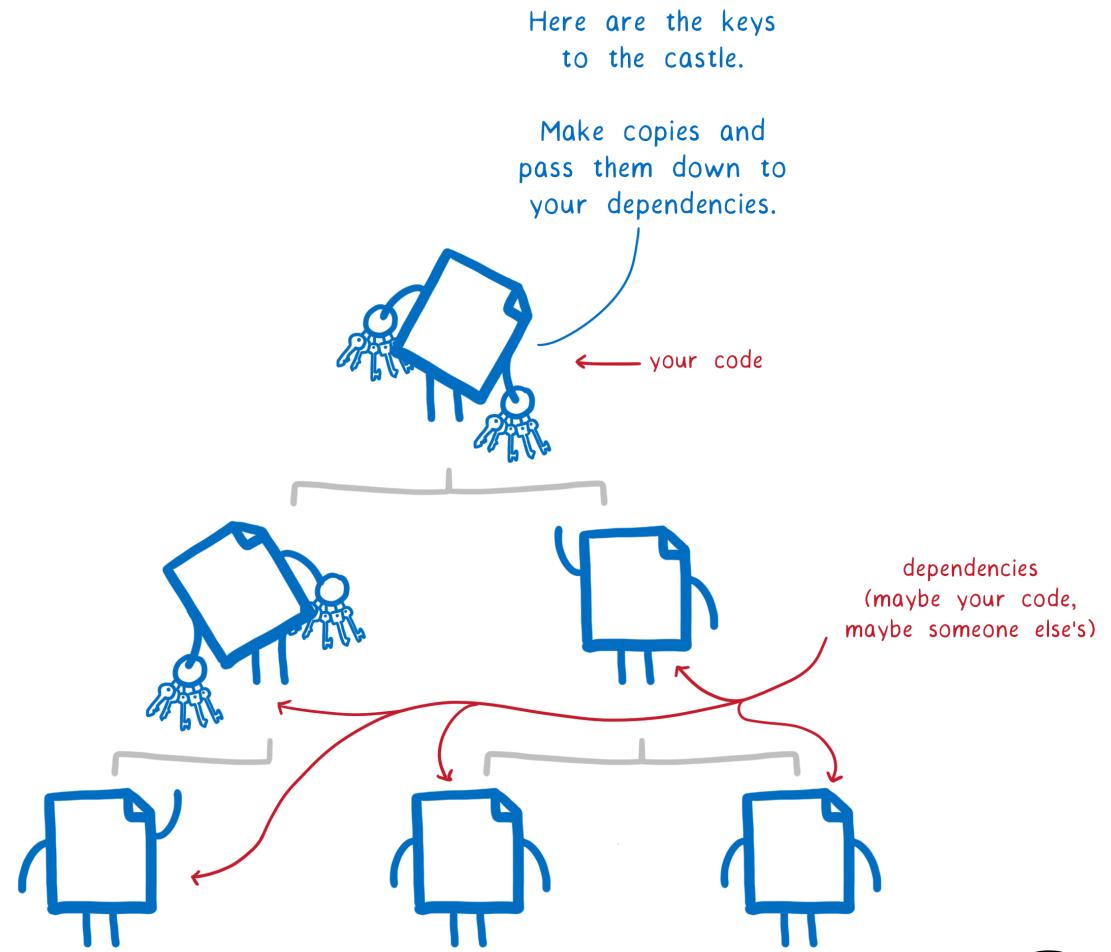
# WASM - What's next?

composition of an  
average code base



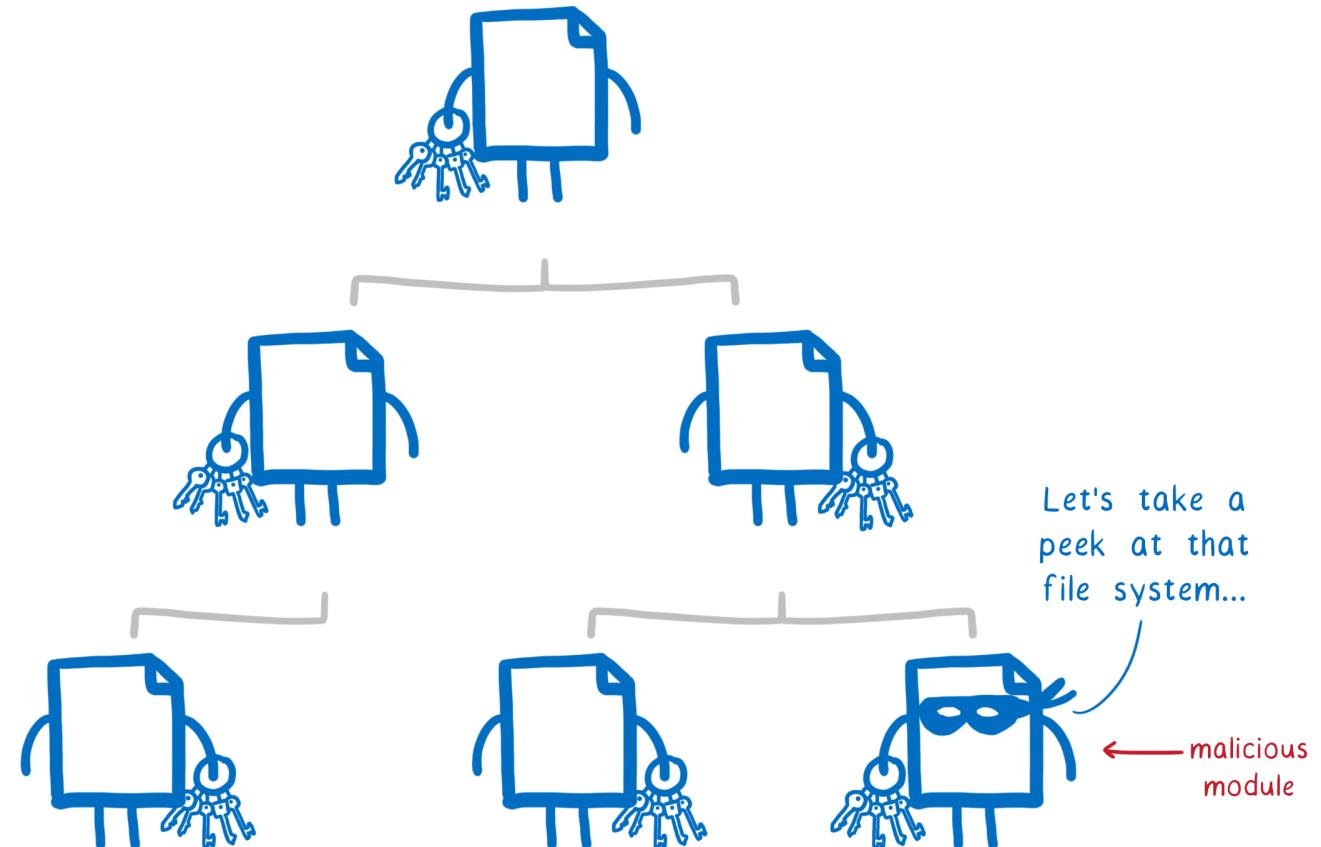
0101  
0101

# Dependencies



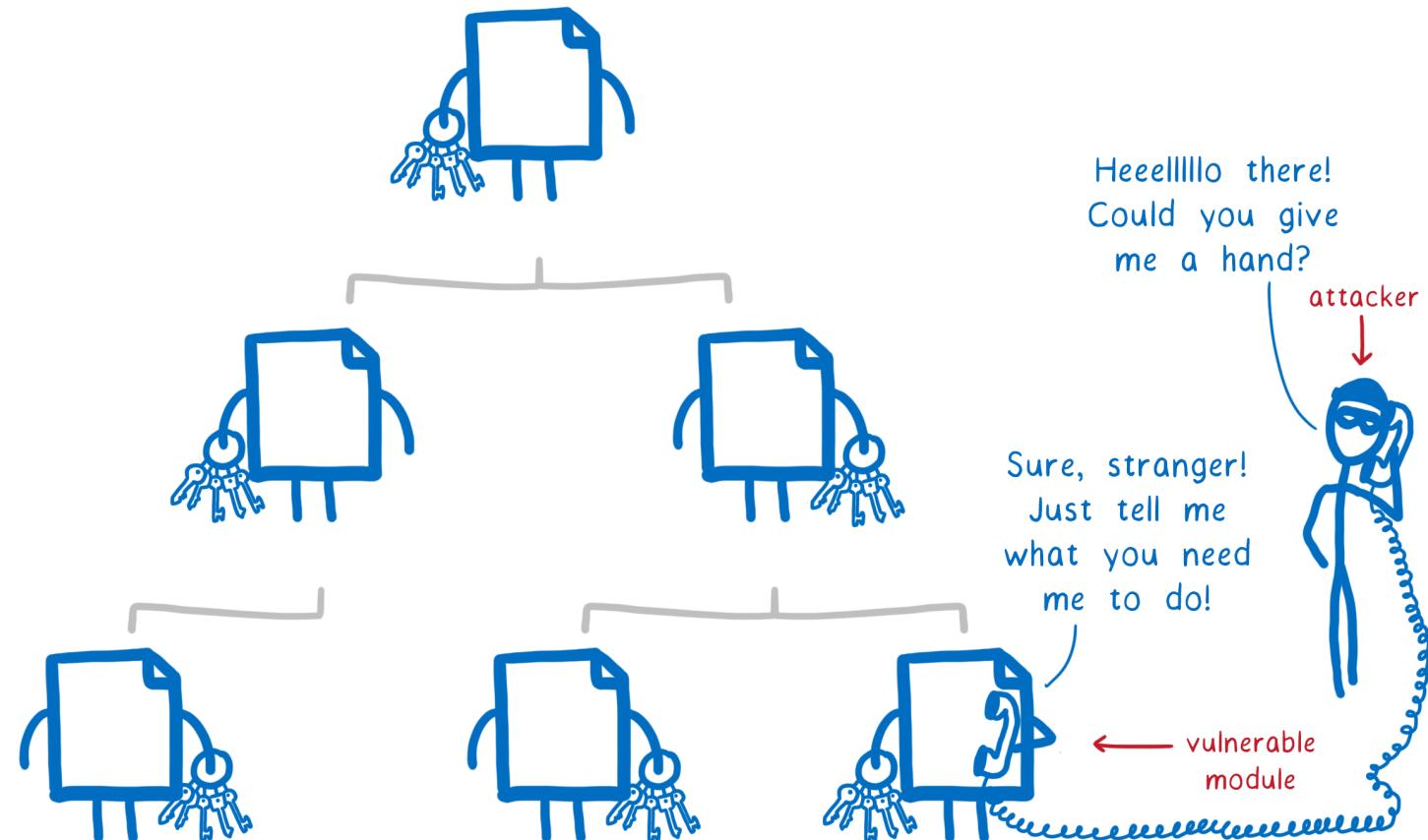
0101  
0101

# Malicious module



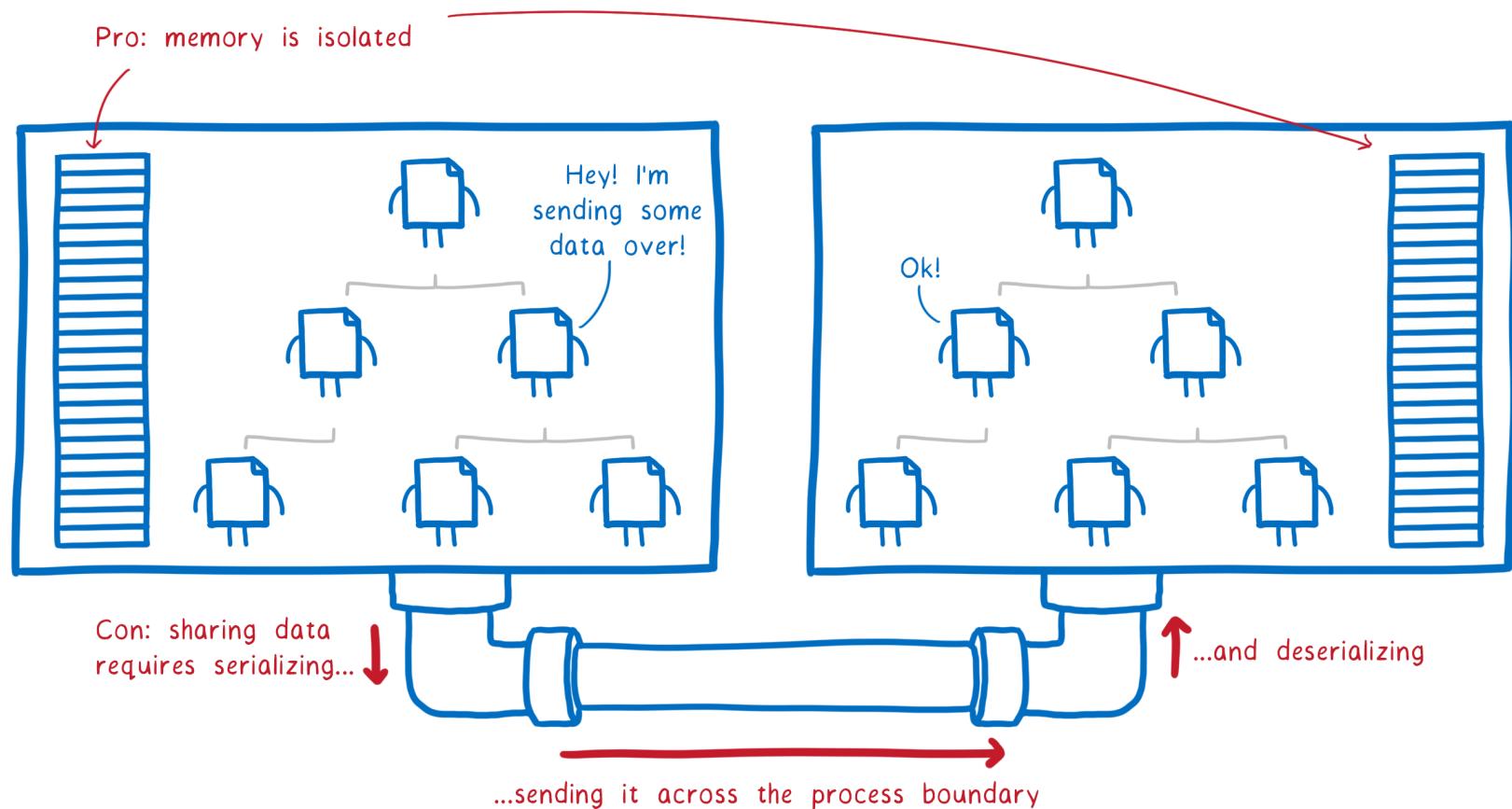
0101  
0101

# Vulnerable module



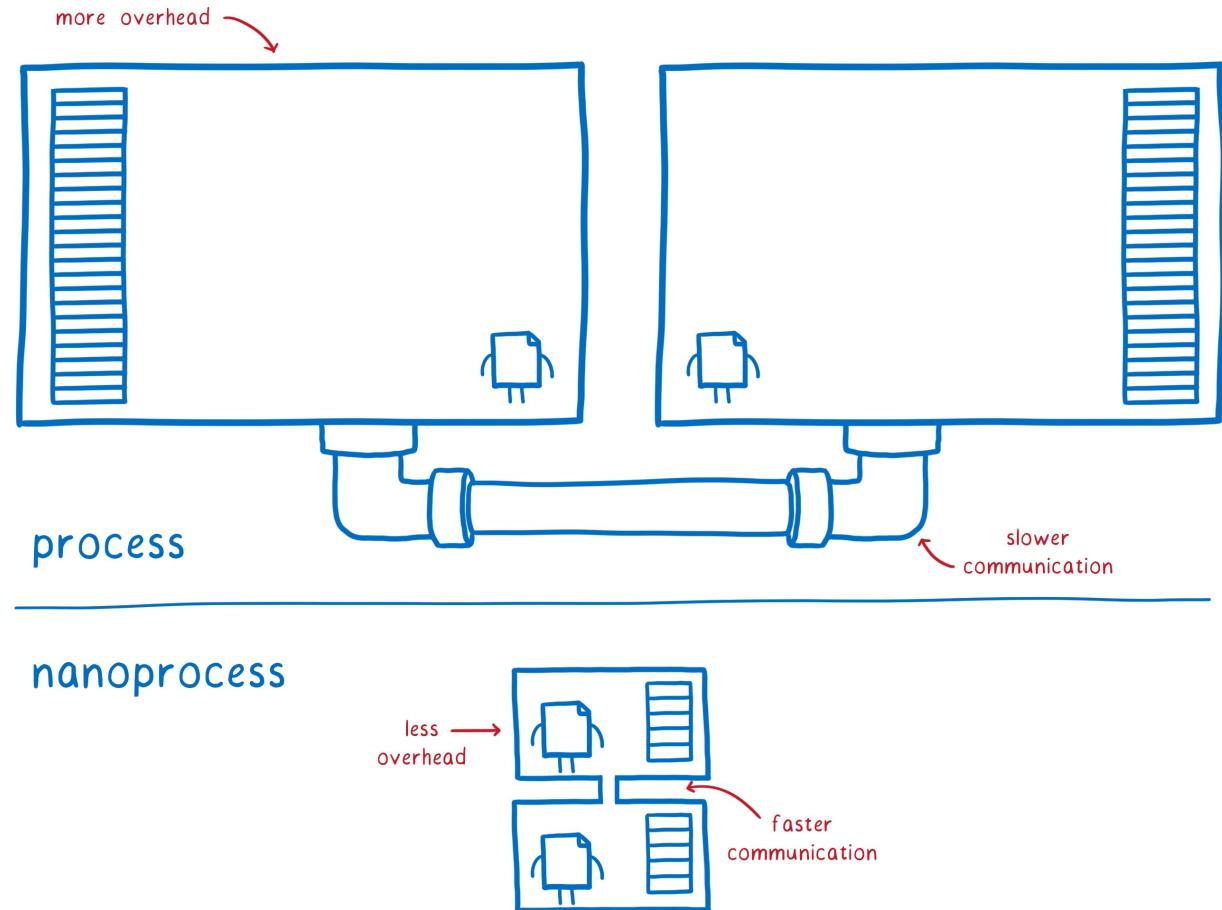
0101  
0101

# Process Isolation



0101  
0101

# WebAssembly Nano-Process

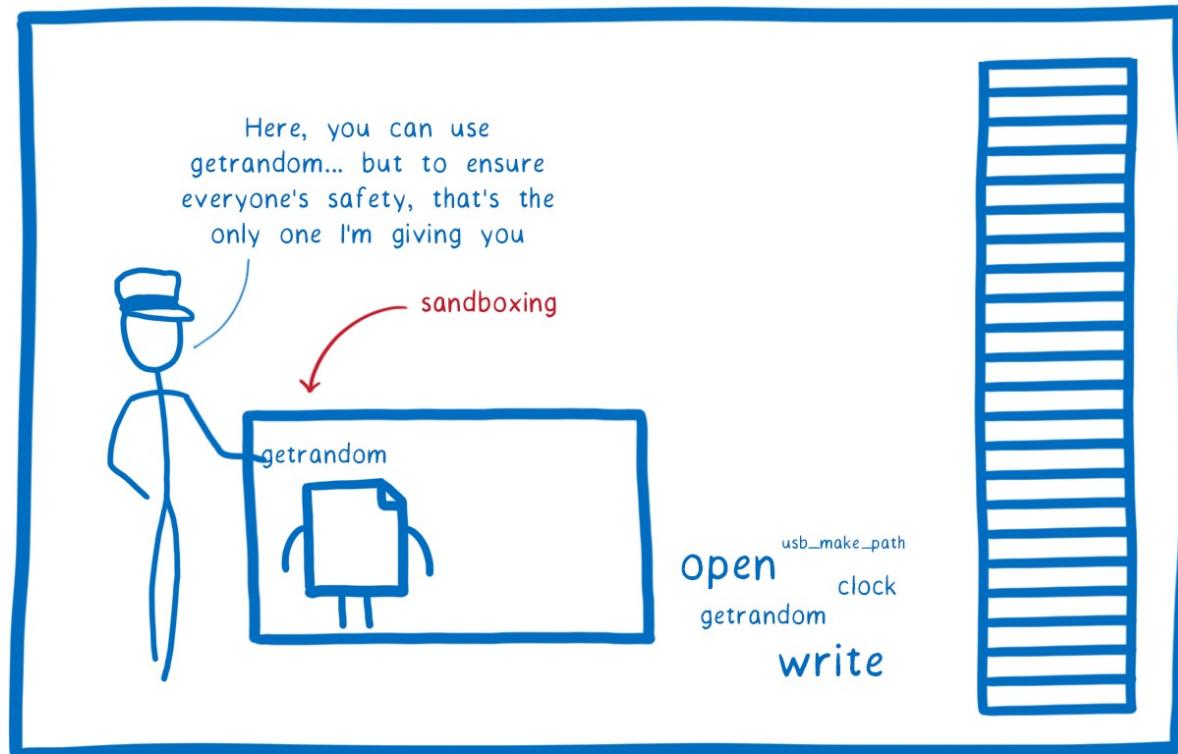


\* not drawn to scale



# WebAssembly Nano-Process

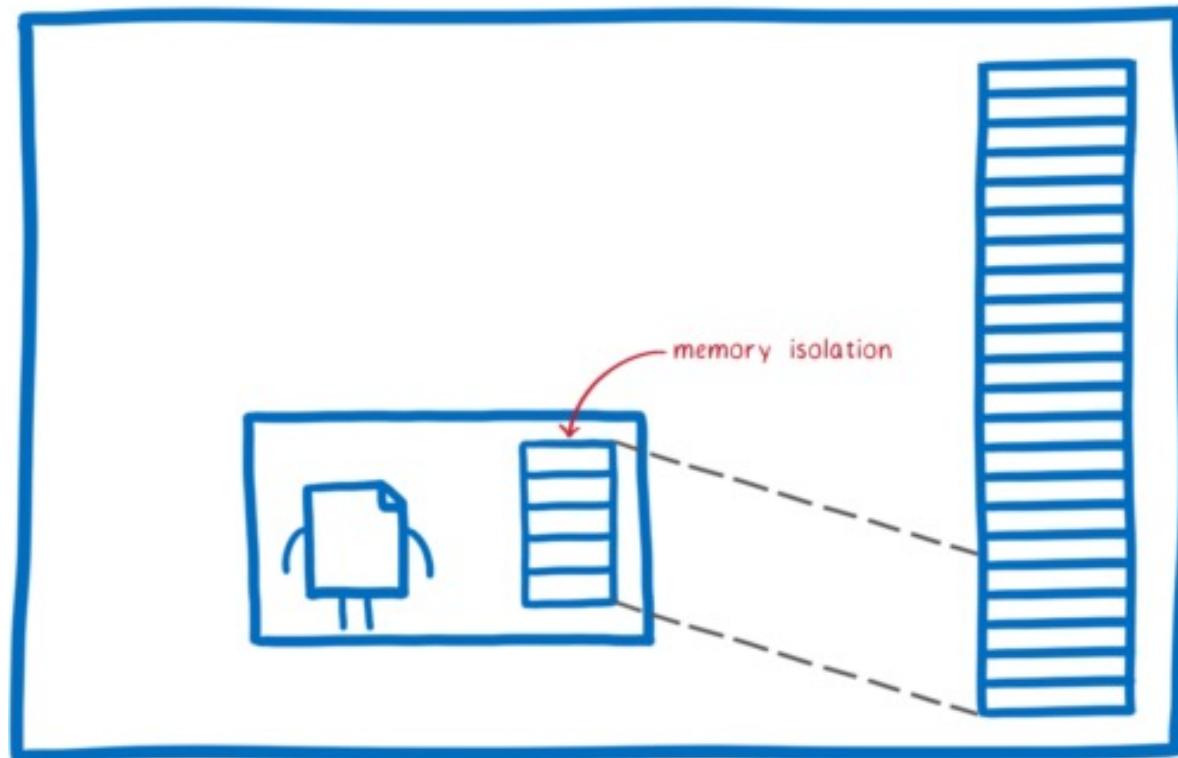
## 1. Sandboxing





# WebAssembly Nano-Process

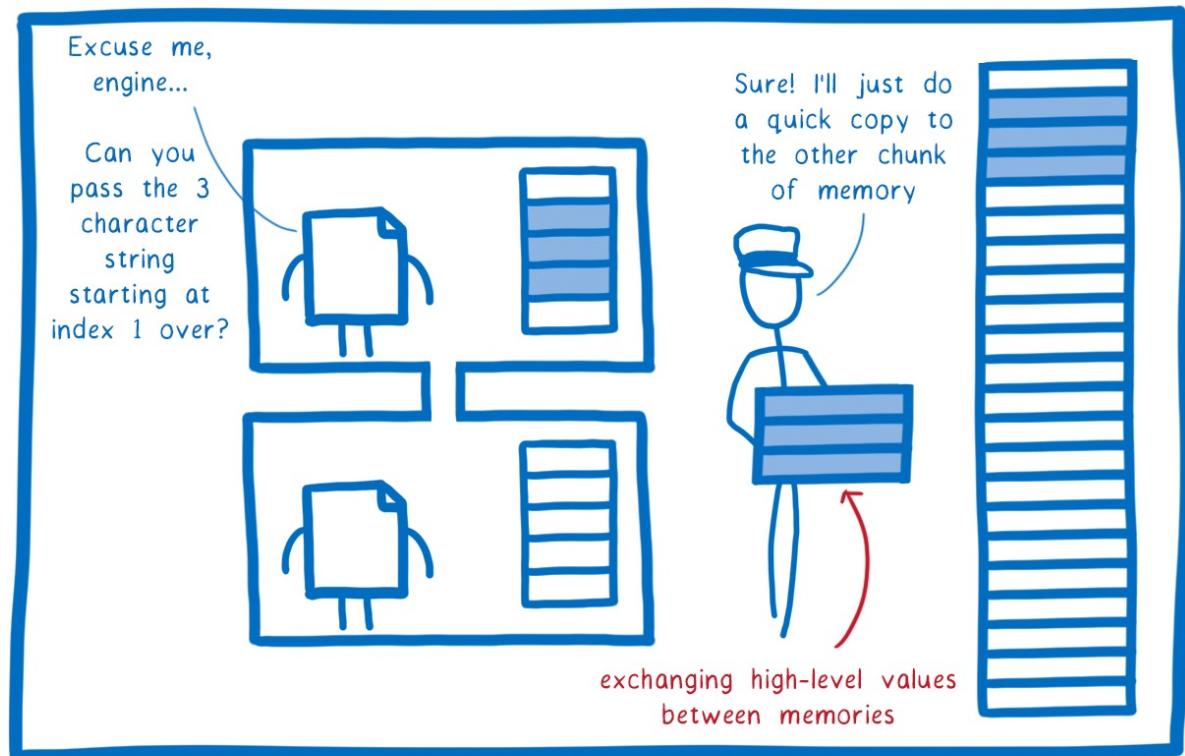
## 2. Memory model





# WebAssembly Nano-Process

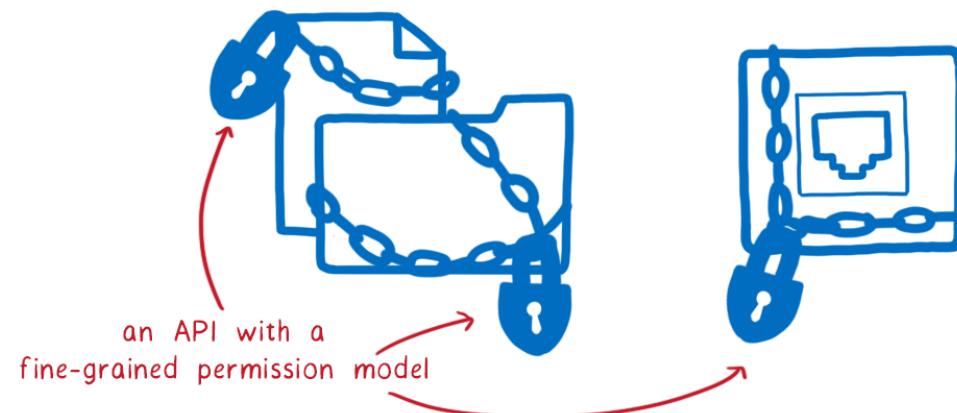
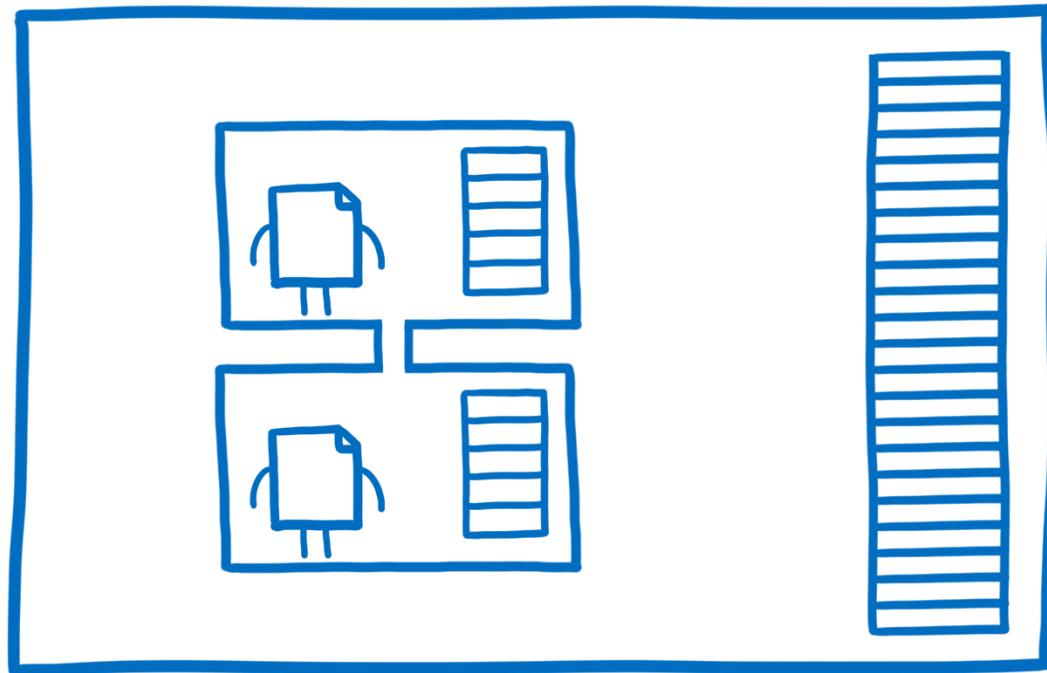
## 3. Interface Types

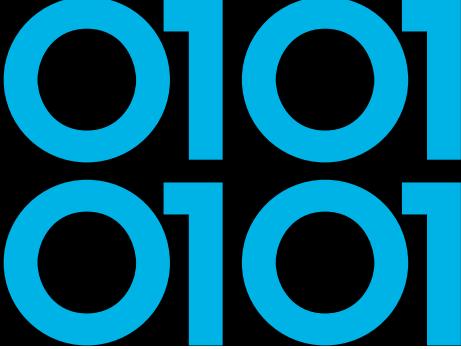




# WebAssembly Nano-Process

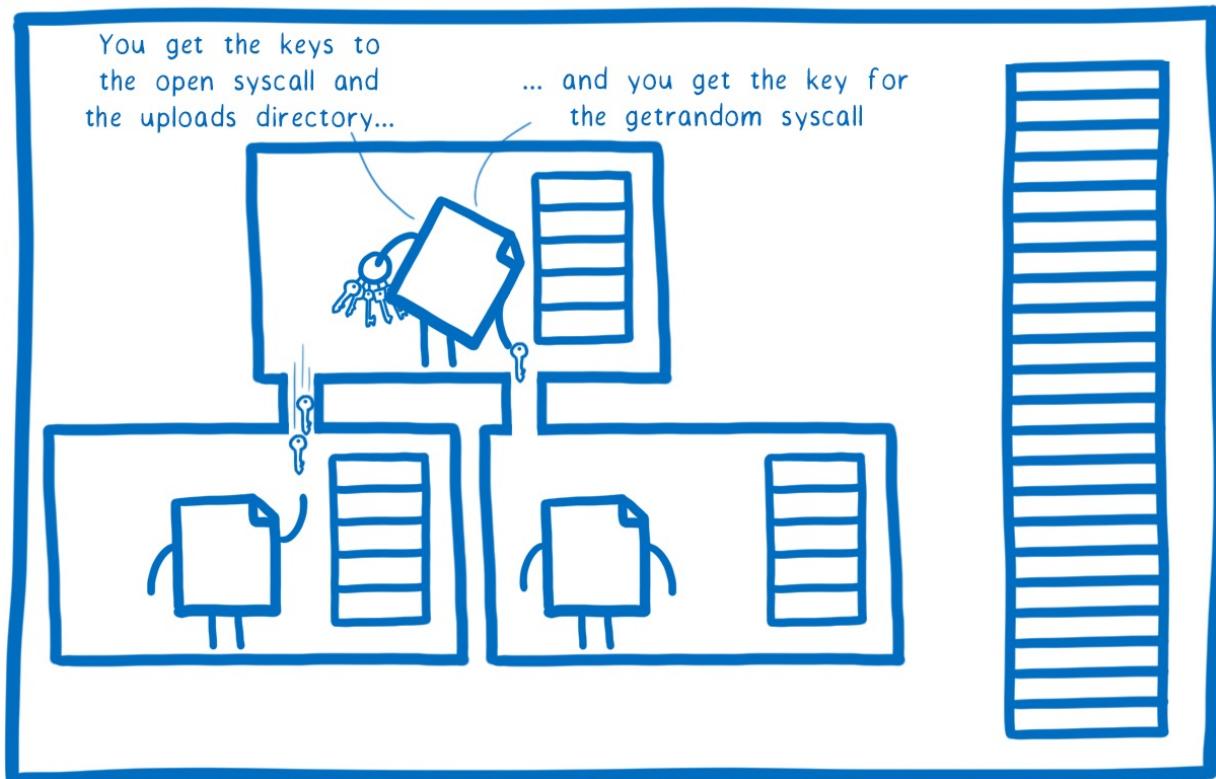
## 4. WebAssembly System Interface





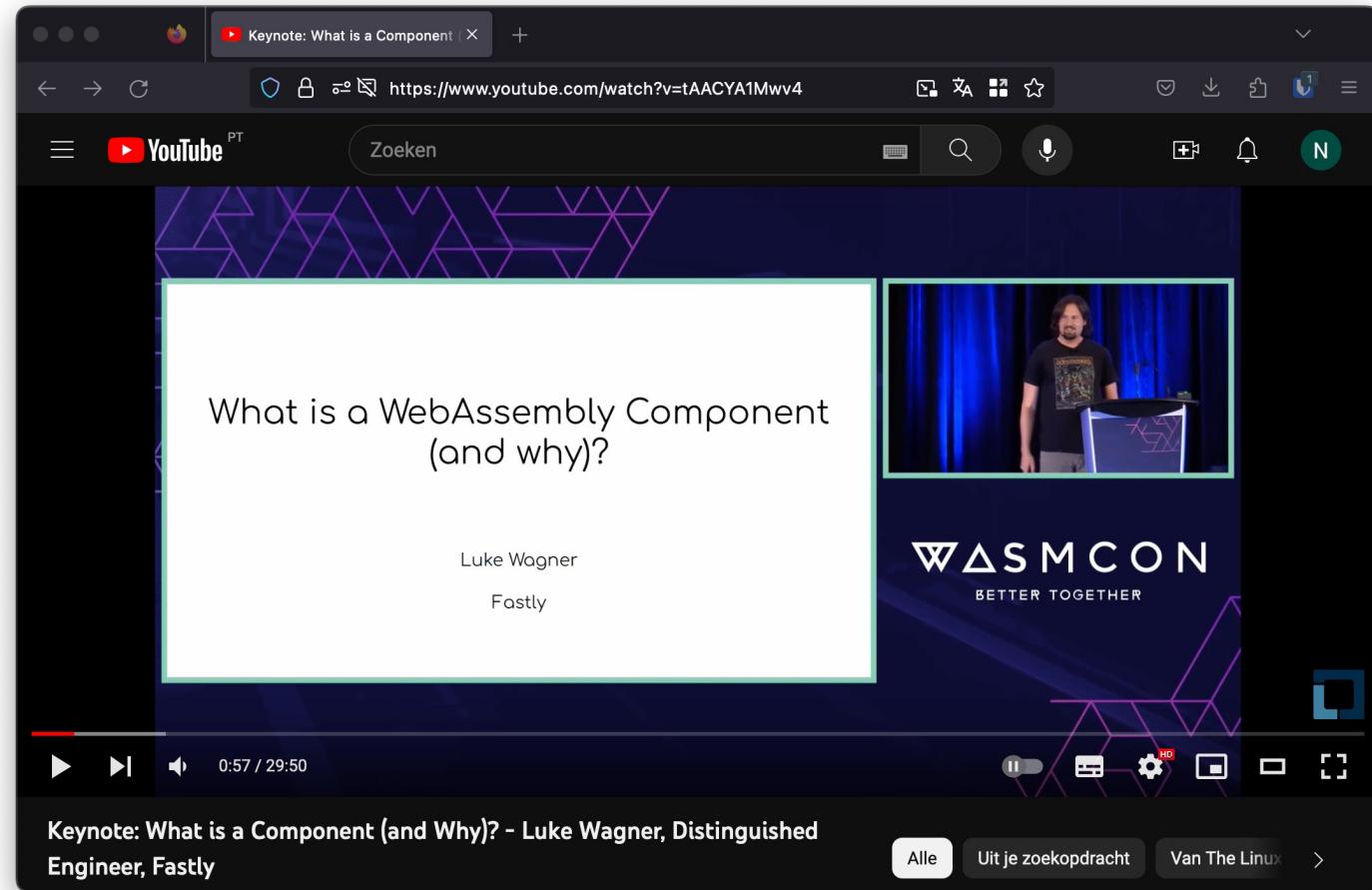
# WebAssembly Nano-Process

## 5. The missing link



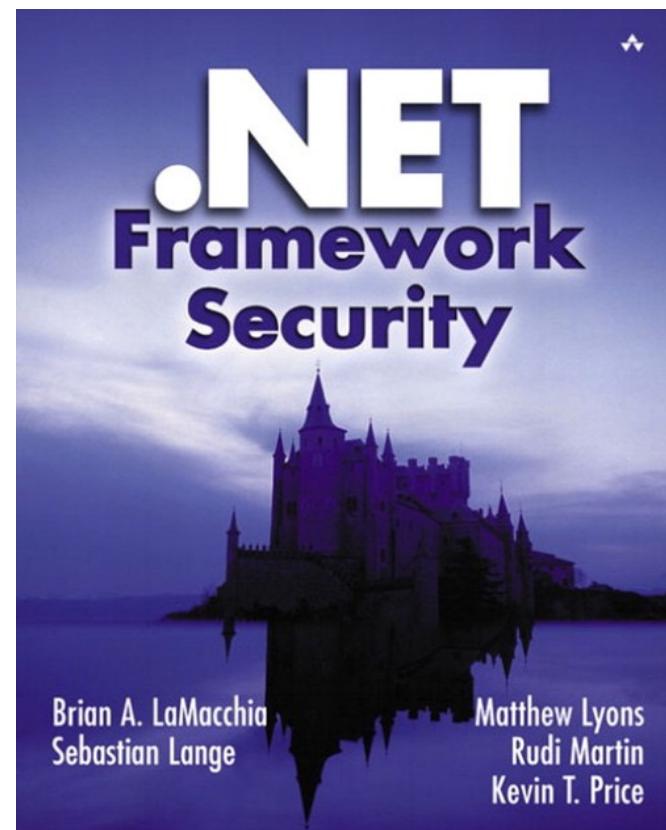
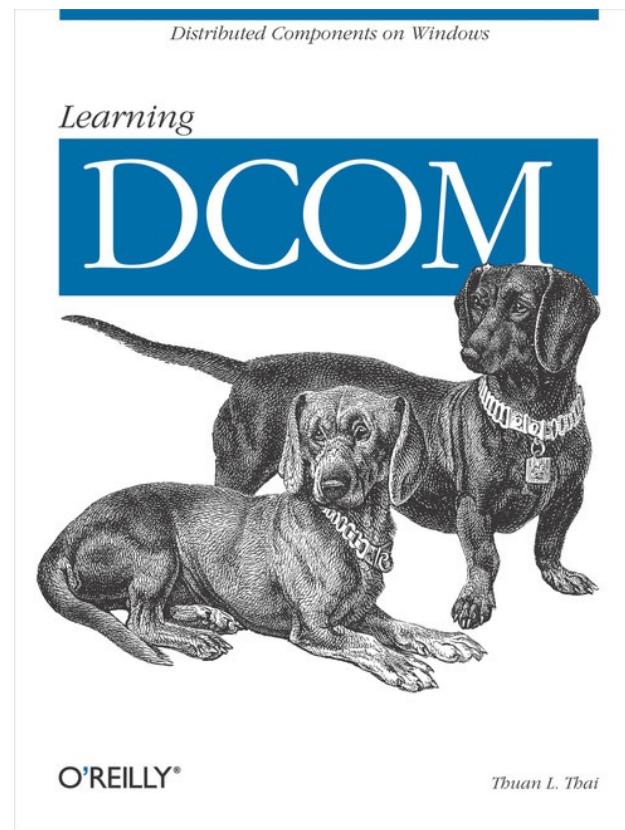
0101  
0101

# WebAssembly Component Model



0101  
0101

# Have we seen this before?



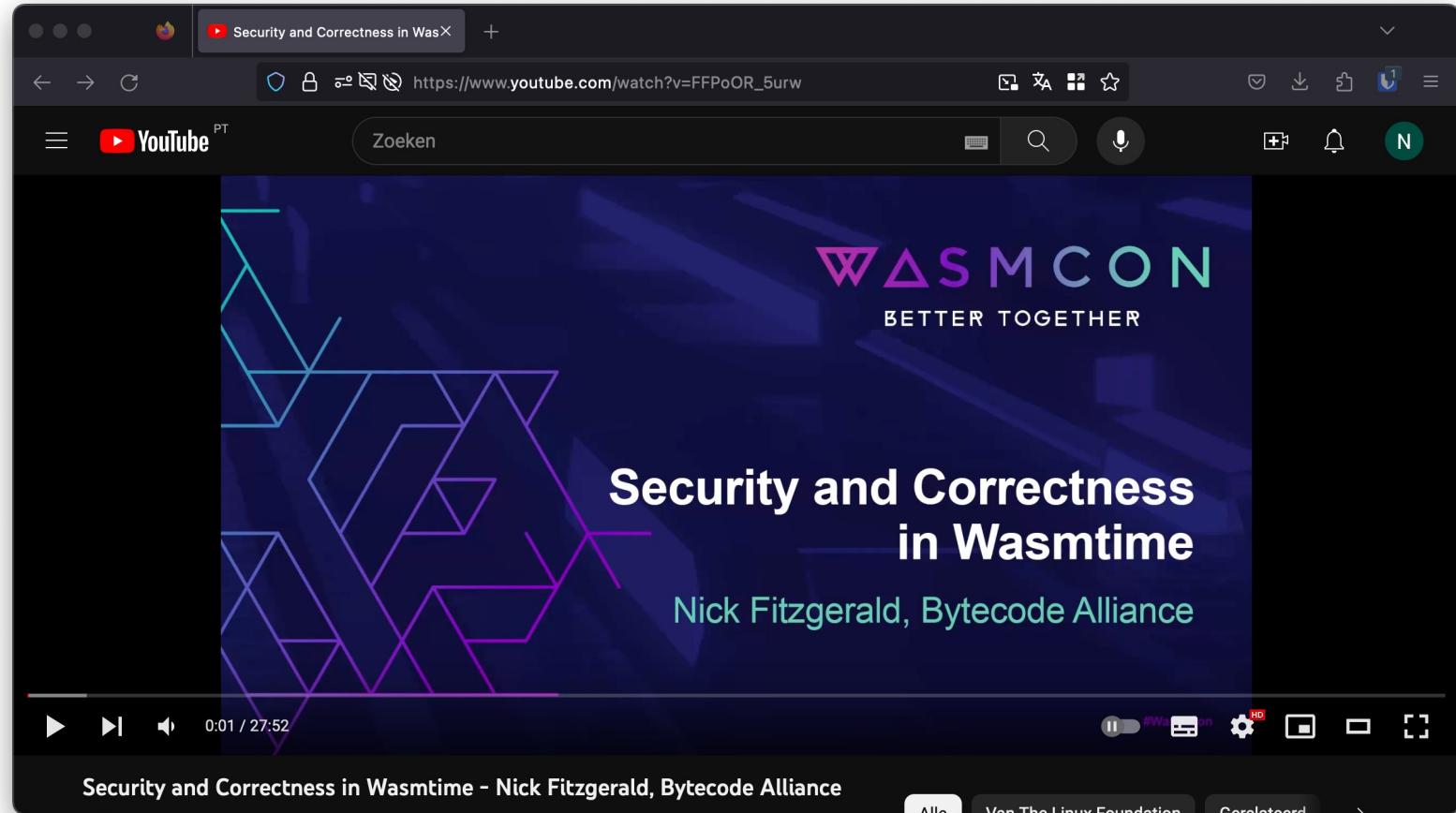


# Runtimes and Security

- 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

0101  
0101

# Runtimes and Security



NDC { Porto }

@nielstanis@infosec.exchange



# 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/influence



# Questions?

- <https://github.com/nielstanis/ndcporto2023-wasm>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Obrigado! Thank you!