What Actually Is WebAssembly
Taking a look under the hood

Caleb Schoepp • Cloud Native Rejekts 2023
Caleb Schoepp
Software Engineer
Fermyon

github.com/calebschoepp
calebschoepp.com
Serverless Apps, powered by WebAssembly
FERMYON

Serverless Apps, powered by WebAssembly.

**SPIN**
Compose serverless Wasm apps quickly.

**FERMYON CLOUD**
Deploy and manage serverless Wasm apps.
Some things you’ve probably heard about WebAssembly
WebAssembly is becoming very popular
WebAssembly 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.”
WebAssembly has an abbreviation

WebAssembly == Wasm
Wasm is being used everywhere

Browsers, server-side, plugins and more
Wasm has major adoption
People are excited about these four properties

1. Security – Sandboxed execution environment
2. Performance – Near native execution speed
3. Polyglot – Supports a wide array of languages
4. Portability - Cross-platform and cross-architecture
Okay, but what actually is Wasm?
People are excited about these four properties

1. Security – Sandboxed execution environment
2. Performance – Near native execution speed
3. Polyglot – Supports a wide array of languages
4. Portability - Cross-platform and cross-architecture
Wasm is another bytecode format

Polyglot

Portability
A Wasm module has two representations
The text format uses s-expressions

```
(module
  (func $add (param $lhs $i32) (param $rhs $i32) (result $i32)
    local.get $lhs
    local.get $rhs
    $i32.add)
  (func $plusOne (param $x $i32) (result $i32)
    local.get $x
    $i32.const 1
    call $add))
```
The most basic Wasm module

(module)
A Wasm module has functions

```wasm
(module
  (func $add (param $lhs i32) (param $rhs i32) (result i32)
    local.get $lhs
    local.get $rhs
    i32.add)
  (func $plusOne (param $x i32) (result i32)
    local.get $x
    i32.const 1
    call $add))
```
Host Runtime

Guest Wasm Module

Function: add

Function: plusOne

calls
Wasm runs on a stack machine
Stack machine example of add(4,3)

```
1  (func $add (param $lhs i32) (param $rhs i32) (result i32)
2    local.get $lhs
3    local.get $rhs
4    i32.add)
```
Wasm lets you export functionality

```
(module
  (func $add (param $lhs i32) (param $rhs i32) (result i32)
    local.get $lhs
    local.get $rhs
    i32.add)
  (export "wasmAdd" (func $add)))
```
And it let's you import functionality

```plaintext
(module
  (import "console" "log" (func $print (param i32)))
  (func $printNumber (param $x i32)
    local.get $x
    call $print))
```
Host Runtime

Import: console.log

imports

Guest Wasm Module

Function: print

Function: printNumber

supplies function

calls

Group of users
Wasm has shared linear memory

```
(module
  (import "console" "log" (func $log (param i32) (param i32)))
  (import "sys" "mem" (memory 1))
  (data (i32.const 0) "Hello, World!\n")
  (func $helloWorld
    i32.const 0
    i32.const 14
    call $log)
  (export "helloWorld" (func $helloWorld)))
```
Security

Host Runtime

Shared Linear Memory

Import: console.log
- reads
- imports
- invokes function
- supplies function

Export: helloWorld
- exports

Guest Wasm Module
- initializes
- Data: "Hello, World!\n"
- Function: log
- Function: helloWorld
- calls

Users
These are just the basics
How does a host runtime execute my Wasm?
The three semantic phases

Decoding 

Validation 

Execution 

Security
Some popular Wasm runtimes

Browser
Wasmtime
Wasm3
Portability
How do I compile my code to Wasm?
Rust has great Wasm support
C/C++ has good support too
Interpreted languages are a little more tricky
You need to compile the interpreter to Wasm
# WebAssembly Language Support Matrix

<table>
<thead>
<tr>
<th>Language</th>
<th>Core</th>
<th>Browser</th>
<th>WASI</th>
<th>Spin SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Python</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Java</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>CSS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>C# and .NET</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>C++</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>TypeScript</td>
<td>✔</td>
<td></td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Ruby</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>C</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Swift</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>R</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Objective-C</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Shell</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Scala (JVM)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Scala (native)</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Go</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>PowerShell</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Kotlin (JVM)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Kotlin (Wasm)</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Rust</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Dart</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
What is WASI and the Component Model?
WebAssembly Component Model
How can I actually use WebAssembly?
SPIN
The framework to compose serverless WebAssembly apps quickly
The quickest way to deploy and manage your serverless WebAssembly apps
Hyper-efficient serverless on Kubernetes, powered by WebAssembly
Quickstart

Go from blinking cursor to deployed serverless app in 66 seconds.

https://developer.fermyon.com/spin/quickstart
Thank you!

Spin Quickstart