

🌙 MoonBit & WebAssembly 🚀

Hongbo Zhang

Char Scientist@MoonBit

WASMIO 2025



Today's Topic 🌐

1. 🤔 What is MoonBit and Why
2. 🌐 MoonBit + Wasm for the Browser
3. ☁️ MoonBit + Wasm for the Server using Component Model
4. ✨ MoonBit beyond Wasm

About Me

- Creator of ReScript, contributor to OCaml, Flow
- Unique experience: implemented most parts of the whole toolchain for ReScript in the first version
- Paranoid about *compilation performance*!

What does rescript compiler give up by compiling so fast?






zeroexcuses

Feb '23

The rescript compiler appears to compile faster than anything with similar type complexity, i.e. Rust, Jsoo, Scala, Haskell. (esbuild feels faster, but esbuild mostly just strips away TS type signatures).

What is the rescript compiler giving up by compiling so fast? What is being sacrificed ?

What is WebAssembly?

- A binary instruction format for a stack-based virtual machine
 - Introduced by major browser vendors since 2017
- Key features:
 - Near-native performance 
 - Portable across platforms (Write once, run everywhere)
 - Compact encoding (Small size )
 - Sandboxed execution 
- Huge potential but hasn't fully taken off yet

The Problem with Existing Languages 🤪

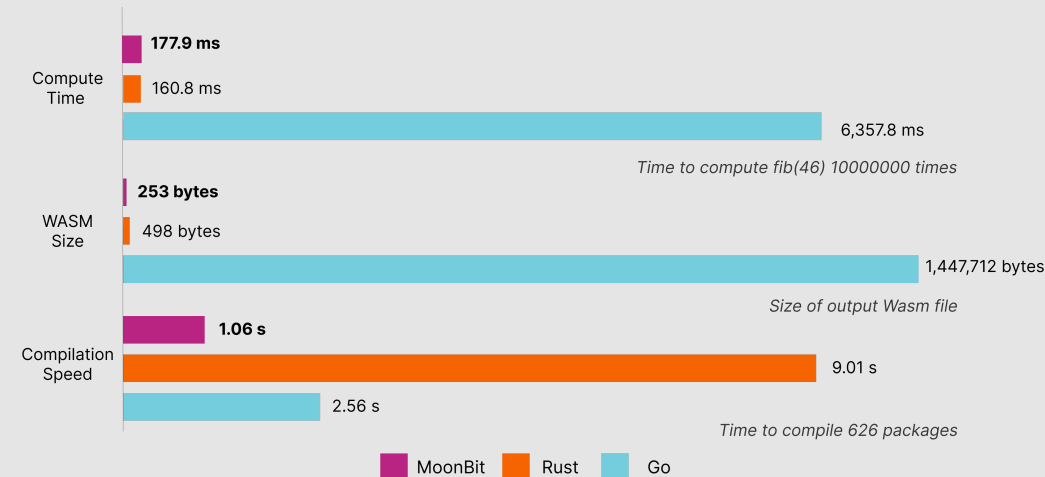
*Almost all major programming languages support wasm but ,
even "Hello World" could generate big wasm output*

The Language Struggle is Real

Language	Problem	Feels Like...
Go	1.8MB "Hello World"	Bringing a tank to a bicycle race
Rust	Learning curve	Climbing Everest in flip-flops
JavaScript	Slow performance	Racing a snail... and losing

Introducing MoonBit! 🌙✨

- A programming language that *designed for* WebAssembly
 - Tiny binary size 🤞
 - Blazing runtime speed 🔥
 - Lightning compilation ⚡
- Focused on dev experience



Key Features of MoonBit

- **Wasm optimization:** small and fast
 - Size in bytes, not megabytes
- Inspired by Rust
 - Generics, type inference, traits
 - More expressive pattern matching
- Rust + GC - complexity for 10-100x faster compilation

Powerful MoonBit Pattern match beyond ADT

```
match data {  
  [{ "age": Number(age), "name": String(name), .. }, ..]  
    => "Hello \{name}. You are \{age}"  
  _ => "not match"  
}
```

- Support pattern match over json, map, resizable array with exhaustive checking
- Custom pattern support coming soon

Developer Experience matters 🎮

- IDE co-designed with the language on day 1
 - Parallel and fault-tolerant type checking
- Out of box debugging
- Real-time responsiveness
- **Cloud IDE:** Zero-install, browser-based with full code intelligence
- Native AI completion support

Demo

MoonBit + Wasm for Browser

- **Performance bottlenecks** in JavaScript-heavy apps
- Great story in FFI with JavaScript
 - WasmGC tight integration
 - Take advantage of JS builtin string proposal

Example: `print_and_concat`

```
pub fn print_and_concat(a : String, b : String) -> String {  
    let c = a + b  
    println(c)  
    c  
}
```

- 182 bytes 🙌
- MoonBit string is JS String for Wasm, zero copy
- Reuse the browser's garbage collector, easy to debug with Chrome devtools

DOM Manipulation: The Easy Way

```
extern type DOM // mapped to externref in Wasm

fn set_css(self : DOM, key : String, value : String) -> Unit = "dom" "set_css"

pub fn change_color(dom : DOM) -> Unit {
    dom.set_css("color", "red") // oo style
}
```

JS glue code

```
const { instance } = await WebAssembly.instantiateStreaming(
  fetch(
    new URL("../dom.wasm", import.meta.url)
  ),
  {
    dom: {
      set_css: (dom, key, value) => {
        dom.style[key] = value;
      },
    },
    ...
  }
);
```

- No bind gen, [MoonBit+WasmGC Repo](#)

Palindrome example: both performance and elegant

Unicode safe Palindrome

```
// generated by ChatGPT
function isPalindrome(normalizedStr) {
  if (!normalizedStr) return true; // Empty strings are palindromes
  let left = 0;
  let right = normalizedStr.length - 1;
  while (left < right) {
    const leftCP = normalizedStr.codePointAt(left);
    const rightCP = normalizedStr.codePointAt(right);
    if (leftCP !== rightCP) {
      return false;
    }
    left += leftCP > 0xFFFF ? 2 : 1;
    right -= rightCP > 0xFFFF ? 2 : 1;
  }
  return true;
}
```

MoonBit panlindrome: Efficient and elegant

```
pub fn palindrome(p : String) -> Bool {  
  loop p.view() {  
    [head, .. middle, tail] =>  
      if head == tail {  
        continue middle  
      } else {  
        return false  
      }  
    [_] | [] => true  
  }  
}
```

- $O(n)$ complexity, stack allocated and unicode safe

Real-world example



Consuming A High Performance Wasm Library from JavaScript



In [one of our previous blog posts](#), we have already started exploring the use of JavaScript strings directly within MoonBit's Wasm GC backend. As we have previously seen, not only is it possible to write a JavaScript-compatible string-manipulating API in MoonBit, but once compiled to Wasm, the resulting artifact is impressively tiny in size.

In the meantime, however, you might have wondered what it will look like in a more realistic use case. That is why we are presenting today a more realistic setting of rendering a Markdown document on a JavaScript-powered web application, with the help of the MoonBit library [GopherJS](#) and [WasmJS](#) (GopherJS Building Blocks).

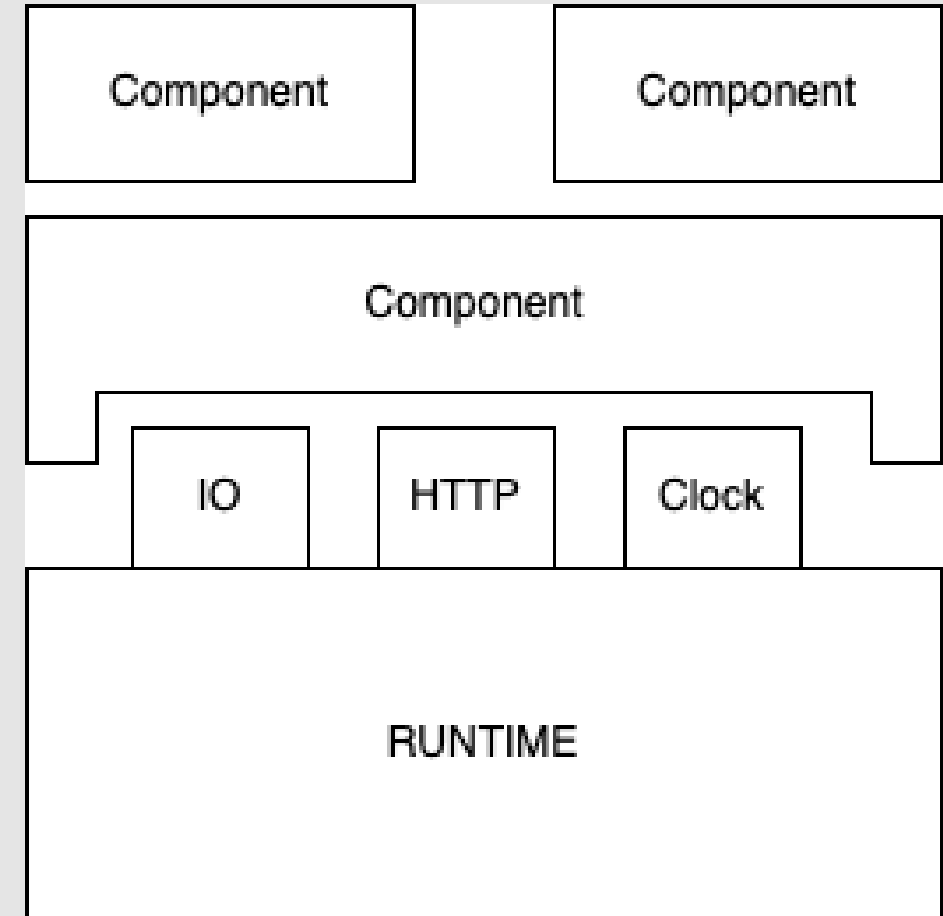
MoonBit + Wasm for Server

Frequently Asked Questions

- Wasm does not support IO,
 - How to read files in MoonBit?
 - How to write servers in MoonBit?
- ...

Component Model to the Rescue!

- Define standardized FFI (CLI, HTTP, Crypto, etc.) using WIT
- Enables language-agnostic/platform-agnostic composition
- Better modularity for cloud & edge computing
- MoonBit supports both Wasm linear and WasmGC



MoonBit vs. Other Languages

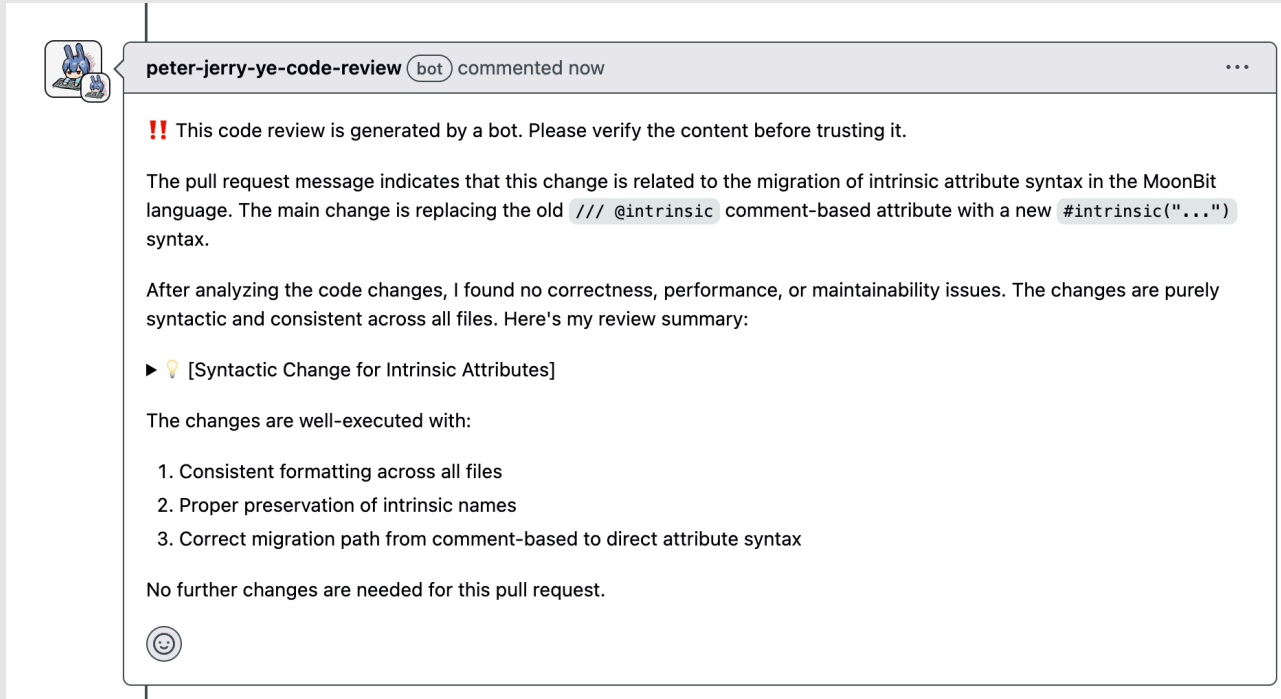
A simple HTTP server returning "Hello World"

Language	Binary Size (KB)
MoonBit	27
Rust	100
Python	17408

Way smaller than containers!

```
pub fn handle(  
    _request: @types.IncomingRequest,  
    response_out: @types.ResponseOutparam  
) -> Unit {  
    try {  
        let response: @types.OutgoingResponse = @http.response!(200)  
        guard response.body() is Ok(outgoing_body)  
        response_out.set(Ok(response))  
        guard outgoing_body.write() is Ok(outgoing_stream)  
        @io.println_sync!("Hello World!", stream=outgoing_stream)  
        outgoing_stream.drop()  
        guard outgoing_body.finish(None) is Ok(_)   
    } catch { _ => () }
```

Real-world Example: MoonBit Powered Microservices



Code review agent for MoonBit using MoonBit + Component Model

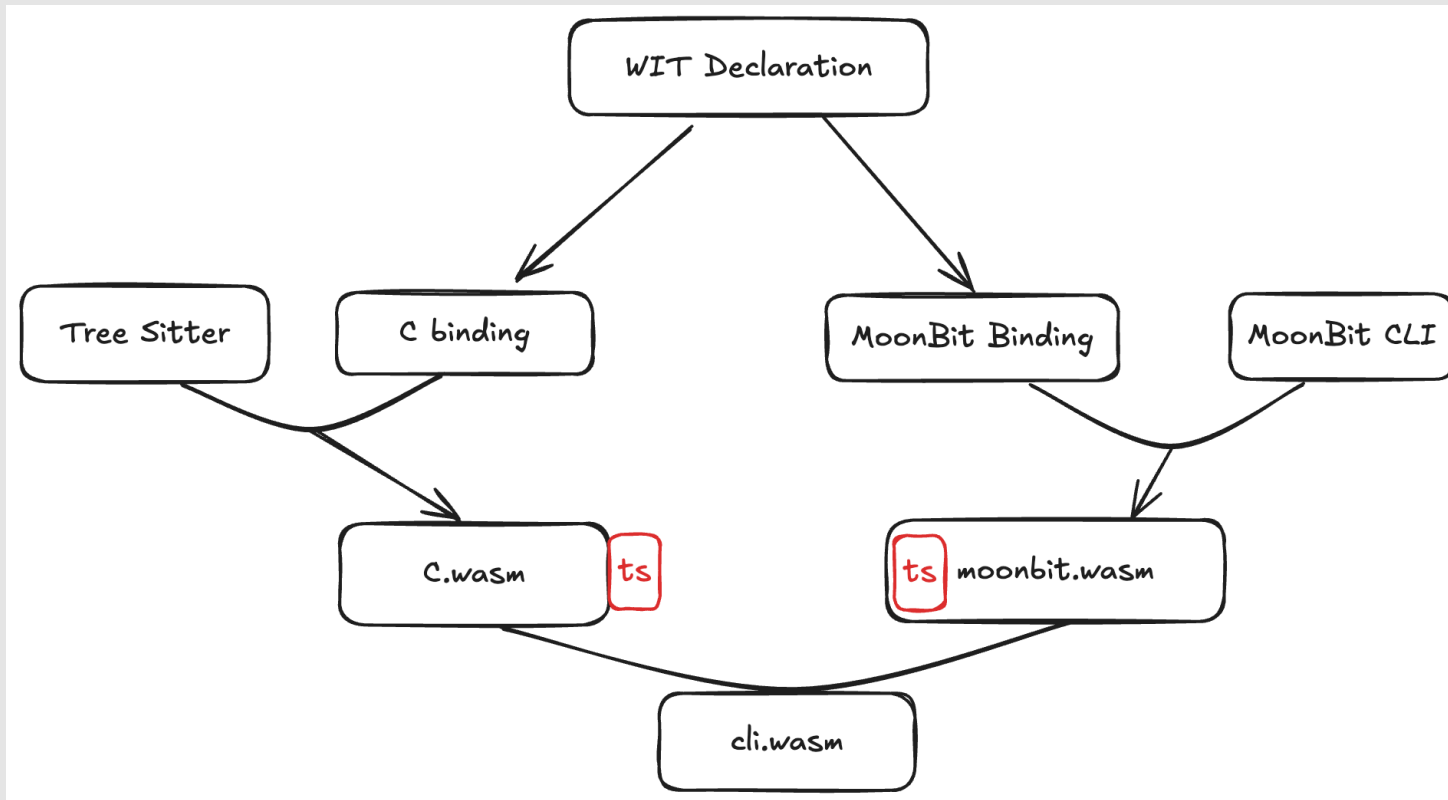
MoonBit Supports Component Model

Arbitrary Interface

1. Define interface with WIT
2. Generate bindings using `wit-bindgen`
3. Develop your application logic
4. Componentize using `wasm-tools component`
 - `embed --encoding utf16` : Embed WIT file and meta information
 - `new` : Transform to component
5. Run with `wasmtime` or `jco`

MoonBit + Component Model for More Possibilities

- Reuse existing libraries



MoonBit + Component Model for More

- Define interface

```
package peter-jerry-ye:tree-sitter@0.1.0;
// C exports, MoonBit imports
interface language {
    use types.{language};
    get-language: func() -> language;
}
interface types {
    resource language { }
    resource parser {
        constructor();
        set-language: func(language: borrow<language>);
    }
}
```

MoonBit + Component Model for More

- Provide binding in C and build with `wasi-sdk`

```
exports_own_language_t
exports_get_language(void) {
    exports_language_t *language =
        malloc(sizeof(exports_language_t));
    language->language = tree_sitter_json();
    // transform to resource
    return exports_language_new(language);
}
// ...
```

MoonBit + Component Model for More

- Use generated library in MoonBit

```
pub fn run() -> Result[Unit, Unit] {  
    let language = @language.get_language()  
    let parser = @types.parser()  
    parser.set_language(language)  
    let tree = parser.parse_string(...)  
}
```

- And compose with `wac plug moonbit.wasm --plug c.wasm -o exe.wasm`
<https://github.com/bytecodealliance/wac>

MoonBit + Component Model in the Future

- Come to learn more about MoonBit and component in our workshop tomorrow
- Integrate code generation and `wasm-tools` directly in the toolchain
- Distribute **Phase 5** proposals as ready-to-use libraries
- Implement WASIp3 (futures and streams) support
- Expand ecosystem with more Component Model-based libraries

Beyond Wasm

- MoonBit supports multiple backends
 - WebAssembly (both linear and GC)
 - Native
 - Compile to C for MCU, tiny binaries
 - Compile to LLVM bitcode
 - Self hosting in the long run
 - JavaScript
 - The elm architecture in MoonBit
 - beta.mooncakes.io made in MoonBit

MoonBit Community

- MoonBit beta comes out this year
- Moonbit lang on X: @moonbitlang
- Discord Community: <https://discord.gg/5d46MfXkfZ>
- Package manager: mooncakes.io