

# Wasm x WebEditor

Beer Bash 🍺 2022/11/11

Portalチーム 越智 大貴

# 章立て

- なんでWasm（WebAssembly）をテーマにしたのか
- WebAssemblyとはなにか
- 強み
- WebAssemblyが動くまで
- WasmメモリメモリWebEditorの紹介
- 付録
- 出典

# なんでWasm (WebAssembly) をテーマにしたのか

- Goで作ったライブラリ・CLIツールをWebアプリとして公開できないかな...でもわざわざWebAPIは作りたくないし...
  - 確か、GoはWasmにビルドできたような...
- 動いた！**コードが再利用できた**！凄い！
- でもWebAssembly良く知らないな...
- ということでこのテーマにしました
  - (何か動くものを作りたかったので、`x WebEditor` にしました)

# WebAssemblyとはなにか 1

- 前身としてasm.jsがいた
  - JavaScriptのパフォーマンスを向上させるため、Mozilla社が開発
  - C/C++のコードをJavaScriptへトランスパイルすることができる
  - 特定の書き方をすると、ブラウザでの実行までの速度・実行時の速度をはやくできる

# WebAssemblyとはなにか 2

- asm.js -> WebAssemblyへ
  - asm.jsの欠点に対処して利点を最大限に活かすために開発された
  - **ブラウザでネイティブコードに近い速度で実行できる**
  - **C/C++, Rust, Goなどの言語で書かれたコードをWebAssemblyにコンパイルしてブラウザで動作させることが可能**

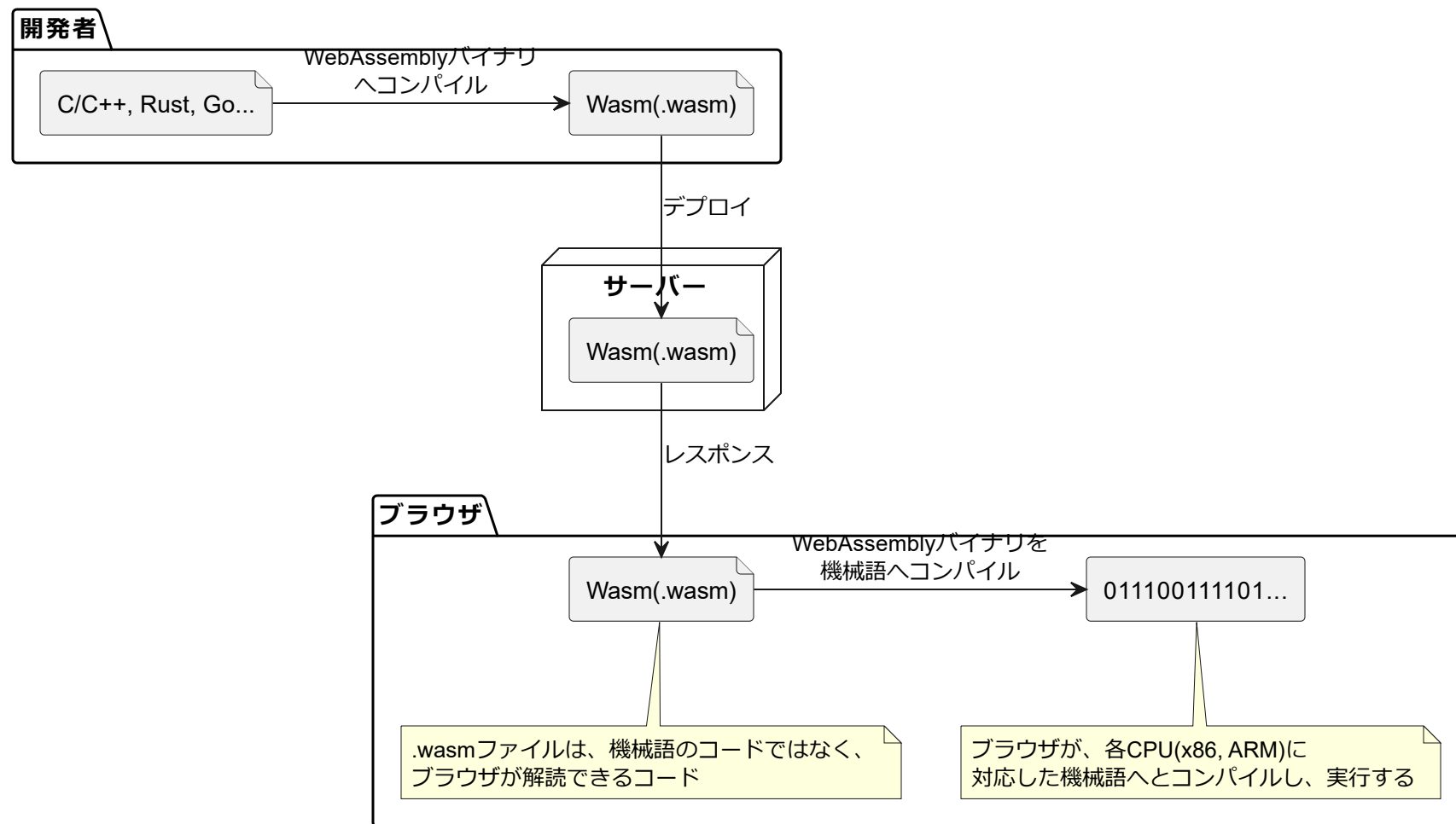
# 強み

- ブラウザでネイティブコードに近い速度で実行できる
- ブラウザでJavaScript以外のプログラミング言語が使える
  - 関連して、コードの再利用ができる（実体験）

# とはいえ

- WebAssemblyがJavaScriptを置き換えるものではなさそうです
  - 単純な処理に、わざわざJavaScript以外の言語でコードを書いてWasmへコンパイルするのは効率が悪そうに思います
  - 実体験としてGo（実際はTinyGo）でコンパイルした.wasmファイルがそこそこサイズあるなと思いました（その他の静的ファイルと比べて）
    - なので処理を高速化したいところを局所的にWasmで作れるといいのかなと思っています
    - ブラウザキャッシュから使えるようにしておくのは対策としていいのかなと思っています（ServiceWorker）

# WebAssemblyが動くまで





# WasmモリモリWebEditorの紹介 1

- <https://dddddddo.github.io/web-editor/>
- 各機能がWasmの処理を呼び出すWebEditorです
  - PHPコードの実行
  - Pythonコードの実行
  - YAMLのフォーマット

# WasmメモリメモリWebEditorの紹介 2

- PHPの実行

## Web Editor



The screenshot shows a web editor interface. At the top left, there is a dropdown menu set to 'PHP' and a 'Run' button. Below these, a code editor contains two lines of PHP code: line 1 is `<?php` and line 2 is `echo "Hello World!";`. At the bottom of the editor, the output 'Hello World!' is displayed.

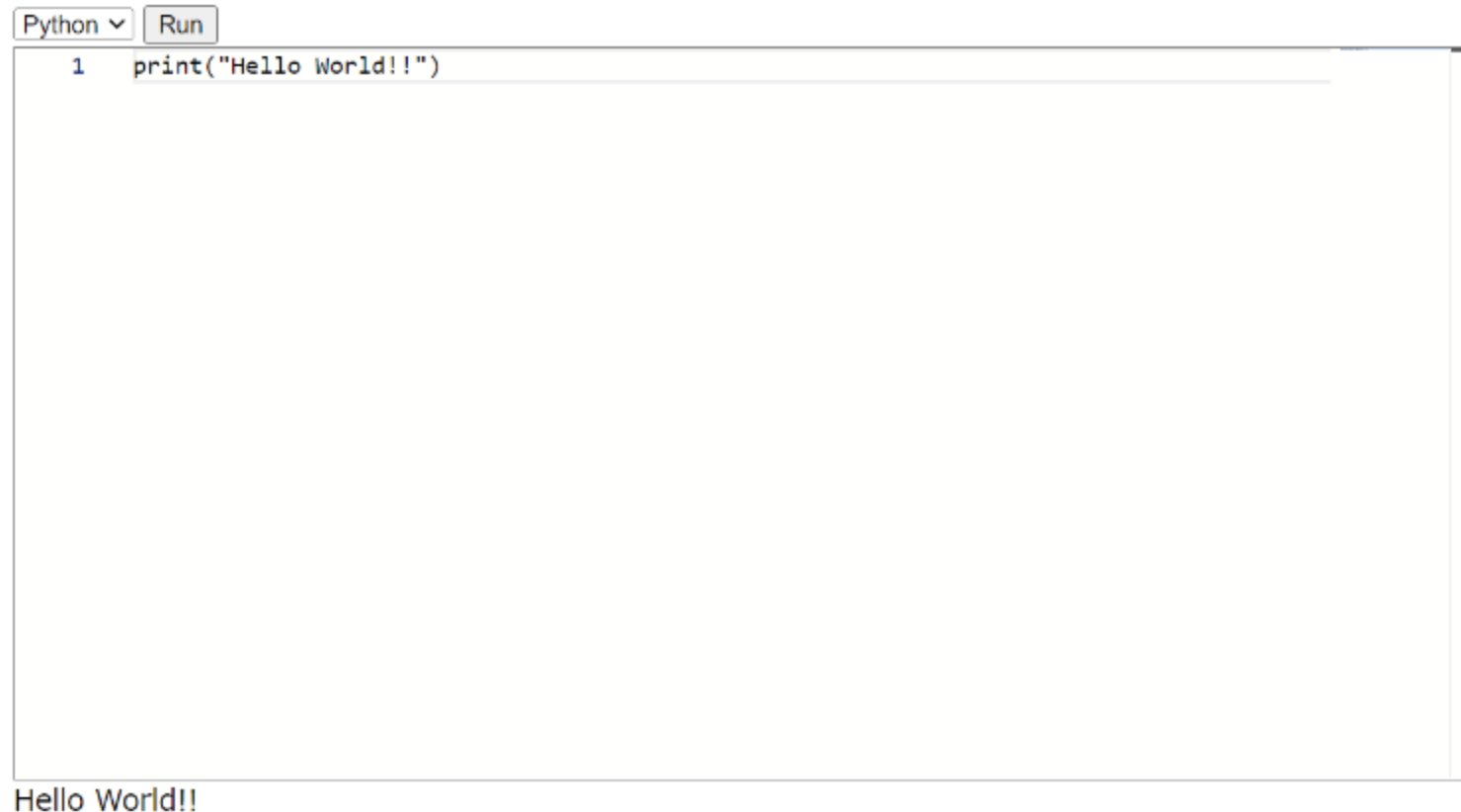
```
PHP ▼ Run
1 <?php
2 echo "Hello World!";
```

Hello World!

# WasmモリモリWebEditorの紹介 3

- Pythonの実行

## Web Editor



The screenshot displays the WasmモリモリWebEditor interface. At the top left, there is a dropdown menu set to 'Python' and a 'Run' button. The main editor area contains a single line of Python code: `1 print("Hello World!!")`. Below the editor, the output 'Hello World!!' is displayed.

```
Python ▾ Run
```

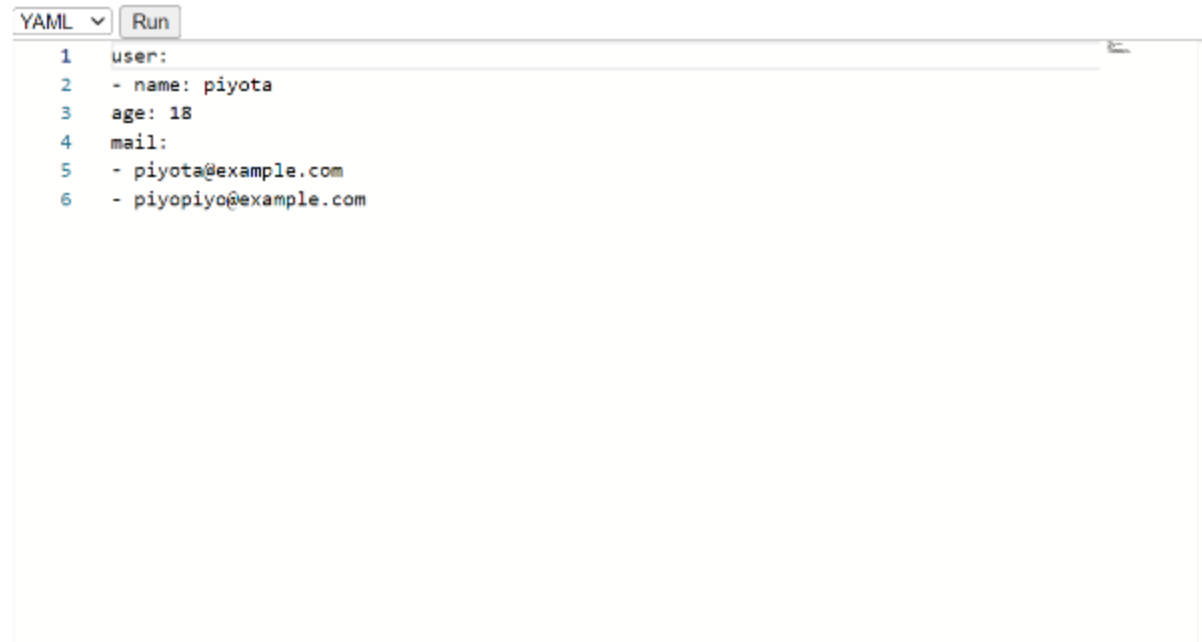
```
1 print("Hello World!!")
```

Hello World!!

# WasmモリモリWebEditorの紹介 4

- YAMLのフォーマット

## Web Editor

A screenshot of the Wasm Mori Mori Web Editor interface. At the top left, there is a dropdown menu set to 'YAML' and a 'Run' button. The main area is a code editor with a light gray background and line numbers on the left. The code is a YAML document with the following content:

```
1 user:
2   - name: piyota
3   age: 18
4   mail:
5     - piyota@example.com
6     - piyopiyo@example.com
```

```
user:
- name: piyota
age: 18
mail:
- piyota@example.com
- piyopiyo@example.com
```

**最後に**

**既存のコードを再利用できるのは便利!!**

# 付録

- WasmメモリメモリWebEditor
  - <https://dddddddo.github.io/web-editor/>
  - 以下は利用したOSS
    - エディタ: <https://github.com/microsoft/monaco-editor>
    - PHPコードの実行: <https://github.com/seanmorris/php-wasm>
    - Pythonコードの実行: <https://github.com/pyodide/pyodide>
    - YAMLのフォーマット: <https://github.com/sosukesuzuki/js-yamlfmt>
- プログラミング言語毎にまとめられたWebAssemblyに関連したリンク集
  - <https://github.com/appcypher/awesome-wasm-langs>
- Wasmで処理する趣味Webアプリ
  - <https://dddddddo.github.io/gtree/>

## 出典

- ハンズオン WebAssembly(2022, オライリージャパン) 第1部