# Setup

install software:

* emscripten
* python3

python3 need install websockets library. pip3 install websockets

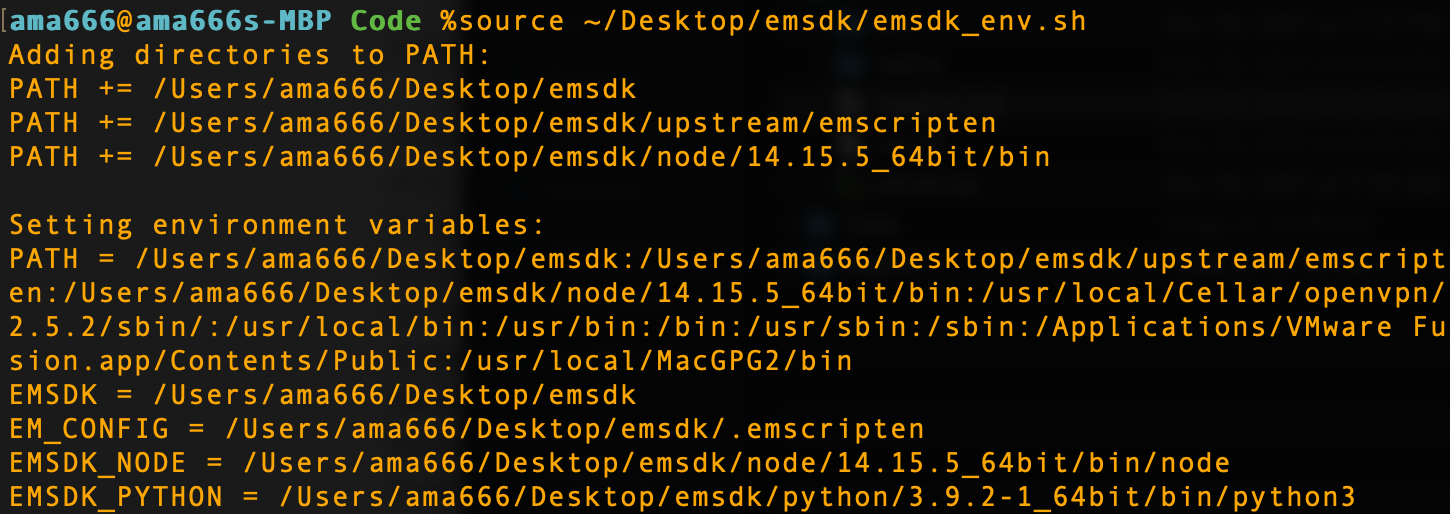
Also this library can be installed via ”python3 -m pip install websockets”

This time I used the python library to set up the server environment instead of using docker to build the server.

# Compile

My Code directory and emsdk directory are stored separately, so I need to import environment variables into the current directory.

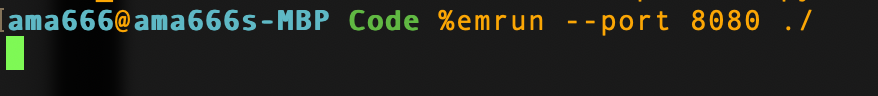
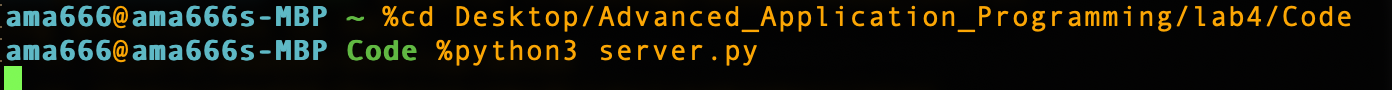
source “emsdk\_PATH”/emsdk\_env.sh



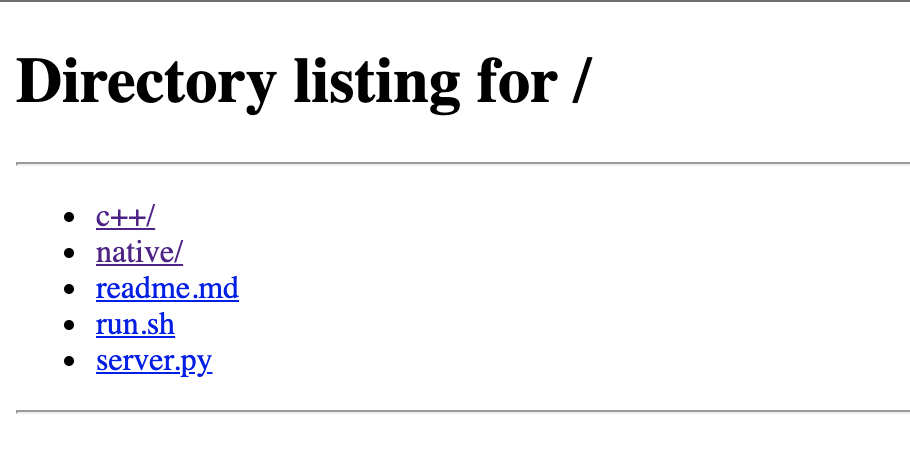
Transcompile command: emcc -lwebsocket -o client.html -v client.cpp

# Run

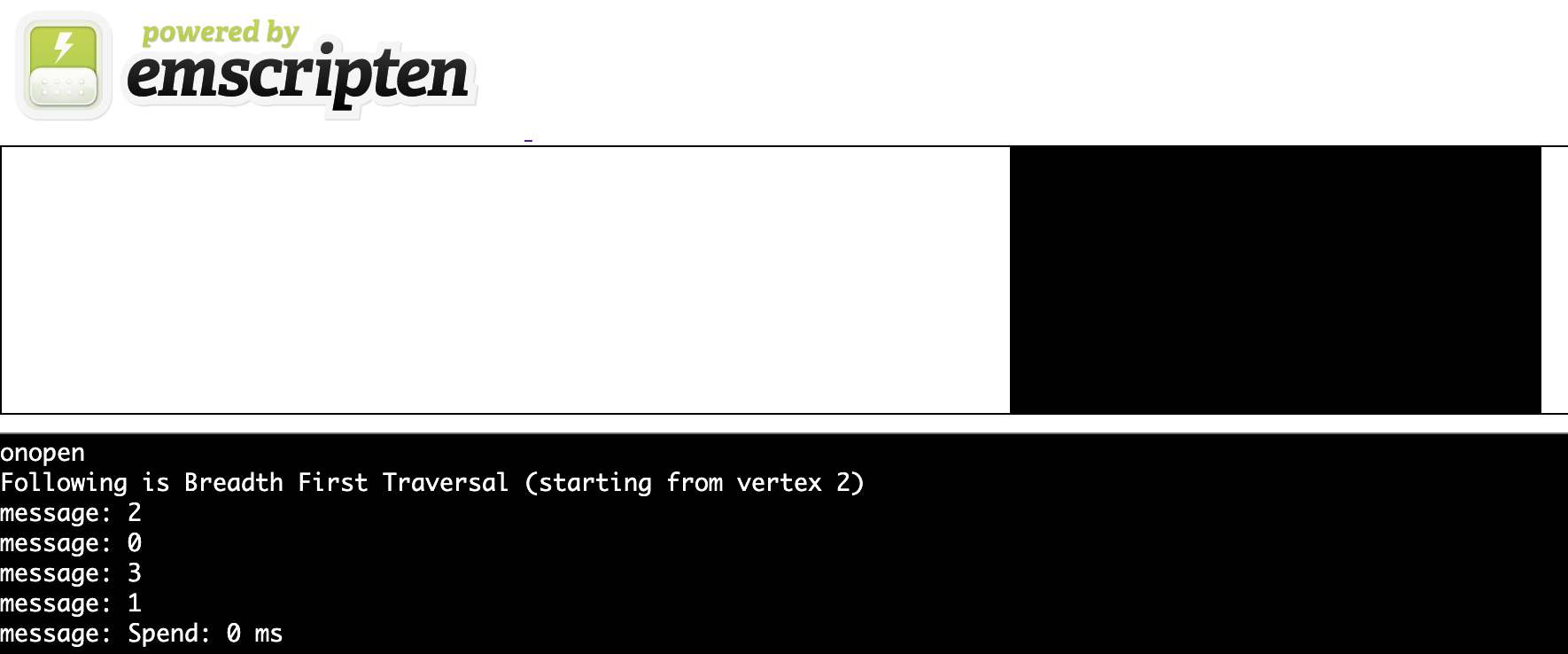
* Start websocket server: python3 server.py
* Start http server: emrun --port 8080 ./
* Open native js implement: http://localhost:8080/native/
* Open c++ implement: <http://localhost:8080/c%2B%2B/client.html>



It will automatically open your browser and the website is like that.

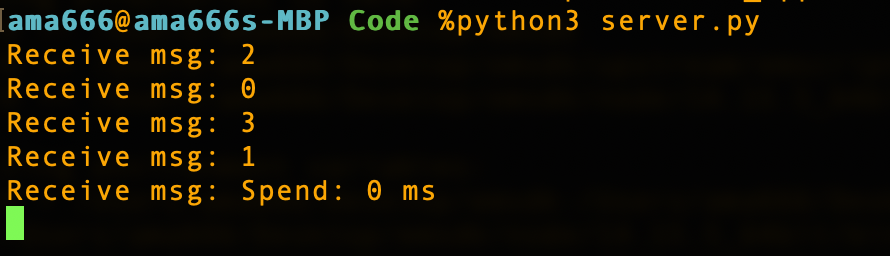


Entry C++ directory and click clint.html.

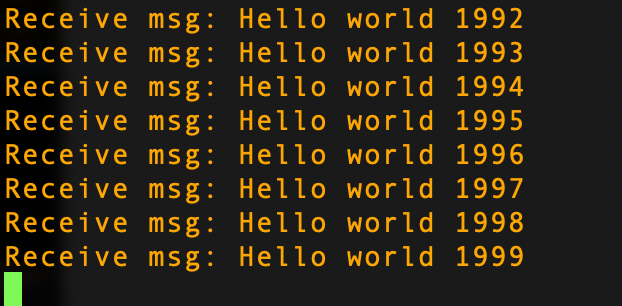


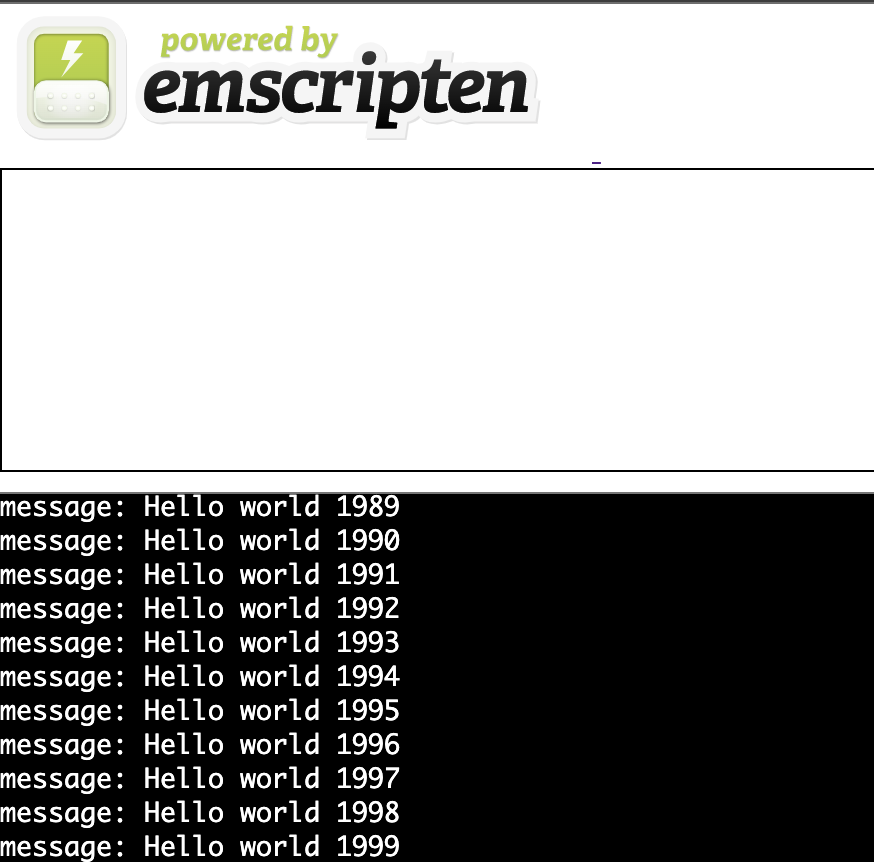
I applied a BFS algorithm on the server transcompiled from C++.

And the native script result.



In order to measure the performance of websocket, I applied a message sending function, which can send "Hello World!" 2000 times to see the speed difference of transcompiled code to native code.





**JS Code**

Send 2000 "Hello world {i}" to web socket server, spend 33.1ms

**Transcompiled Code**

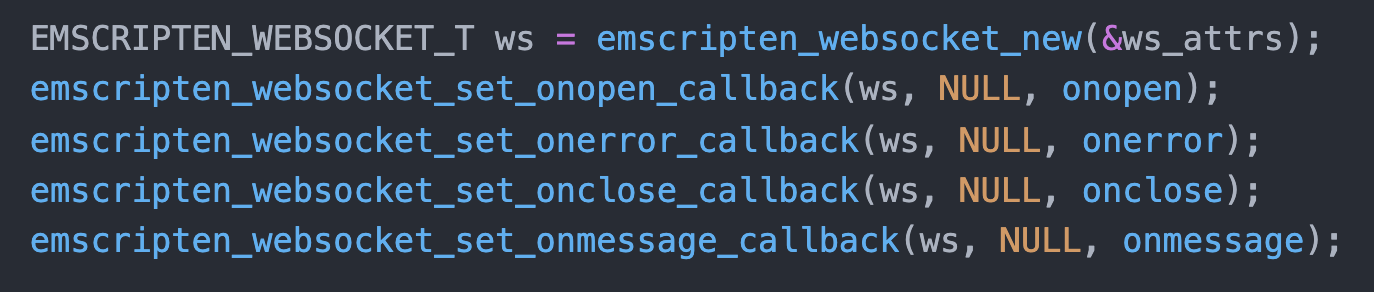
Send 2000 "Hello world {i}" to web socket server, spend 26.5ms;

Combined with wasm, it can speed up the communication speed of websocket.

# Networking Architecture

WebSockets API provides connection-oriented message-framed bidirectional asynchronous networking communication to the browser. It is the closest to TCP on the web that web sites can access, direct access to TCP sockets is not possible from web browsers.

Emscripten provides a passthrough API for accessing the WebSockets API from C/C++ code. This is useful for developers who would prefer not to write any JavaScript code, or deal with the C/C++ and JavaScript language interop. See the system include file <emscripten/websocket.h> for details. One benefit that the Emscripten WebSockets API provides over manual WebSockets access in JavaScript is the ability to share access to a WebSocket handle across multiple threads, something that can be time consuming to develop from scratch.



Many functions are defined in the native C++ code, corresponding to websocket.



# Pros and Cons

Advantages of WebSocket:

* It allows for two-way communication.
* Websockets allow you to send and receive data much faster than HTTP. They're also faster than AJAX.
* Communication between origins (however, this poses security risks).
* Compatibility between platforms (web, desktop, mobile)
* HTTP has a 2000-byte overhead, but WebSocket only has a 2-byte cost.
* Long polling is replaced.
* AJAX calls can only send string data types because WebSockets are data typed.

Cons of Websocket:

* A fully HTML5-compliant web browser is required.
* AJAX-like success mechanisms are not available in Websockets.
* Websockets, unlike HTTP, do not provide intermediary/edge caching.
* It is impossible to employ friendly HTTP statuses, bodies, and other elements to create even a simple protocol of your own.
* HTTP is significantly easier to develop if your application doesn’t take a lot of dynamic interaction.