WebAssembly 1.0 has shipped in 4 major browser engines.
Learn more

Developer’s Guide

This page provides step-by-step instructions to compile a simple program directly to WebAssembly.

Prerequisites

To compile to WebAssembly, some prerequisite tools are needed:

- Git. On Linux and OS X this is likely already present. On Windows download the Git for Windows installer.
- CMake. On Linux and OS X, one can use package managers like `apt-get` or `brew`, on Windows download CMake installer.
- Host system compiler. On Linux, install GCC. On OS X, install Xcode. On Windows, install Visual Studio 2015 Community with Update 3 or newer.
- Python 2.7.x. On Linux and OS X, this is most likely provided out of the box. See here for instructions.

After installing, make sure that `git`, `cmake` and `python` are accessible in PATH. Technically, CMake and a system compiler may not be needed if using a precompiled toolchain, but development options may be a bit limited without them.

Downloading or Compiling the Toolchain

A precompiled toolchain to compile C/C++ to WebAssembly is easily obtained via GitHub.

```
$ git clone https://github.com/juj/emsdk.git
$ cd emsdk
$ ./emsdk install latest
$ ./emsdk activate latest
```

If you are running a Linux distribution for which Emscripten toolchain is not available precompiled (currently Ubuntu 16.04 works best), or if you want to build the whole toolchain from source, Emscripten SDK can also be used to drive the build. The required steps are as follows.

```
$ git clone https://github.com/juj/emsdk.git
$ cd emsdk
$ ./emsdk install --build=Release sdk-incoming-64bit binaryen-master-64bit
$ ./emsdk activate --build=Release sdk-incoming-64bit binaryen-master-64bit
```

After these steps, the installation is complete. To enter an Emscripten compiler environment in the current command line prompt after downloading a precompiled toolchain or building your own, type

```
$ source ./emsdk_env.sh --build=Release
```

This command adds relevant environment variables and directory entries to PATH to set up the current terminal for easy access to the compiler tools.

On Windows, replace `./emsdk` with just `emsdk`, and `source ./emsdk_env.sh` with `emsdk_env` above.

If you are using Visual Studio 2017, `emsdk install` should append with arg `--vs2017`.

Compile and run a simple program

We now have a full toolchain we can use to compile a simple program to WebAssembly. There are a few remaining caveats, however:

- We have to pass the linker flag `-s WASM=1` to `emcc` (otherwise by default `emcc` will emit asm.js).
- If we want Emscripten to generate an HTML page that runs our program, in addition to the Wasm binary and JavaScript wrapper, we have to specify an output filename with a `.html` extension.
- Finally, to actually run the program, we cannot simply open the HTML file in a web browser because cross-origin requests are not supported for the `file` protocol scheme. We have to actually serve the output files over HTTP.

The commands below will create a simple “hello world” program and compile it. The compilation step is highlighted in bold.

```
$ mkdir hello
$ cd hello
$ cat << EOF > hello.c
#include <stdio.h>
int main(int argc, char ** argv) {
  printf("Hello, world!\n");
}
EOF
$ emcc hello.c -s WASM=1 -o hello.html
```

To serve the compiled files over HTTP, we can use the `emrun` web server provided with the Emscripten SDK:

```
$ emrun --no_browser --port 8080 .
```

Once the HTTP server is running, you can open it in your browser. If you see “Hello, world!” printed to the Emscripten console, then congratulations! You’ve successfully compiled to WebAssembly!

GETTING STARTED