



# INTRODUCTION TO WEBASSEMBLY

By Mayur Pillay

## Introduction

What is WebAssembly? WebAssembly, or WASM, is essentially a type of code that allows languages to run on the web. It is the converted file of a programming language that you can use, hence it replaces the programming language file. The major browsers WASM works in is Firefox, Chrome, Safari and Edge and runs via JavaScript.

In this tutorial I will explain the basics of wasm and how to a basic program works with it, using C.

You can download the files from the link below:

<https://github.com/mp-github-acc/WebAssemblyTutorial>

## Getting started

We need a browser, namely one of the previously mentioned that do support WASM.

If you don't have a way to host a local server, be sure to download one, such as XAMPP at <https://www.apachefriends.org/index.html>. Set it up and you will need to open the "index.html" from the GitHub link above (make sure the entire folder is moved with it). The code does have comments to explain the important parts so you can follow along there as well.

We are going to do the coding in <https://webassembly.studio/> and then you will be able to download the files to run it in your browser using the local server.

Let's go!

## Generating files

In WebAssembly Studio, you will have the option to create a new project. Create an Empty C Project. You will see that there is a couple of files that have been generated for you.

The "README.md" and "package.json" files are not relevant. The "build.ts" file is what will generate our wasm file for us. You will also see a "main.c", which is what will be used to generate the wasm file. Then we have the "main.html" and "main.js" files.

You will see the html file does not have much, just a plain page. This is where the results will show, but we will get to that in a bit. The C file is short, just one "main" function that returns a value. The other code in this file is what allows us to access the function when it is converted into a wasm file, as a module. A module is essentially a component of a program.

Select the build and run option. Now we have our "main.wasm" file which is essentially the functionality of the C file in a different format. You will see a bit of code which looks familiar:

```
(func $main (export "main") (type $t1) (result i32)
  i32.const 42)
```

This is our main function from the C file but in a format a bit closer to machine level than human level.

Lastly you will see the bottom right of your screen is a block with a number in it. That is the result of the C function shown on the html via an iframe.

Here is a rundown of what happened:

- The build.ts file compiled the main.c file into main.wasm
- The main.html file is loaded, and since the main.js file is referenced, it loaded too
- The main.js made a request to load main.wasm file (line 1 in main.js)
- An instance of the wasm code was made (lines 2-4)
- A call is made to the main function which returns the value 42  
(Line 5, "instance.exports.main()")
- This value is set to show on main.html  
(Line 5, "document.getElementById("container").textContent")

Hopefully you have a better idea, now we can move on to do a bit more





