WebAssembly

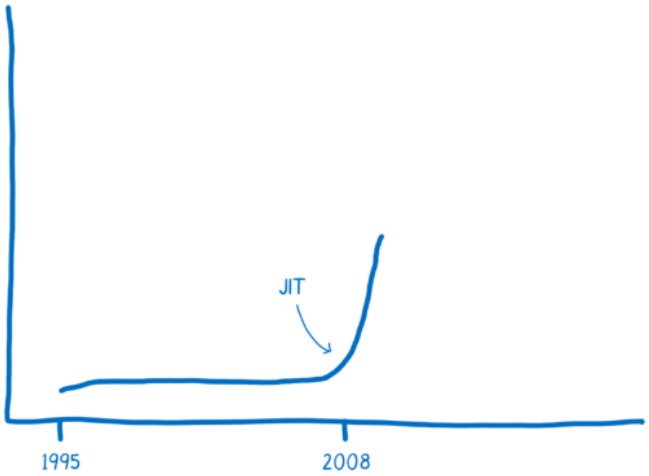
Speaker Qualifications

- Specialize in next-generation technologies
- Author of O'Reilly Videos on Hypermedia, Linking Data, Security and Encryption
- Author of 'Resource-Oriented Architecture Patterns for Webs of Data'
- Teaches and speaks internationally about REST, Semantic Web, Data Science, Machine Learning, GPU Computing, Security, Visualization, Architecture
- Worked in Defense, Finance, Retail, Hospitality, Video Game, Health Care, Telecommunications and Publishing Industries
- International Pop Recording Artist

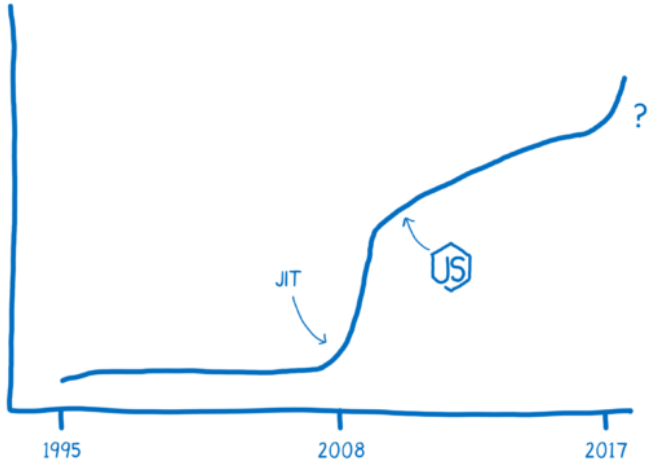
Agenda

- Introduction
- Emscripten
- WebAssembly
- Future

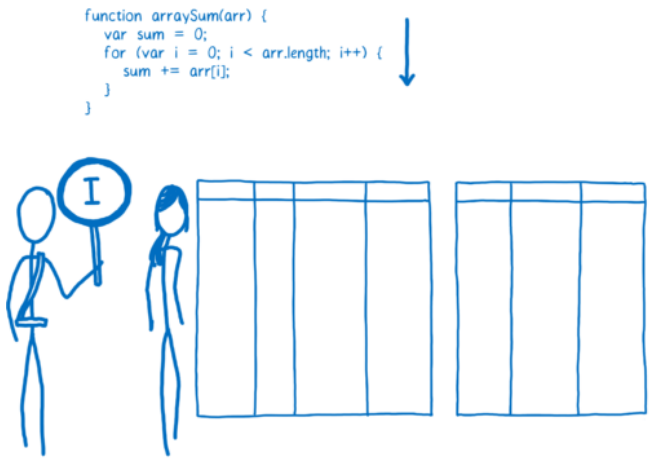
Introduction



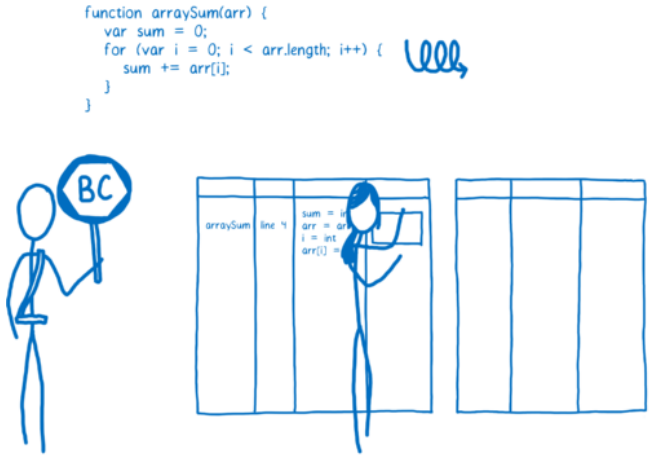
Source: https://hacks.mozilla.org/2017/02/a-cartoon-intro-to-webassembly



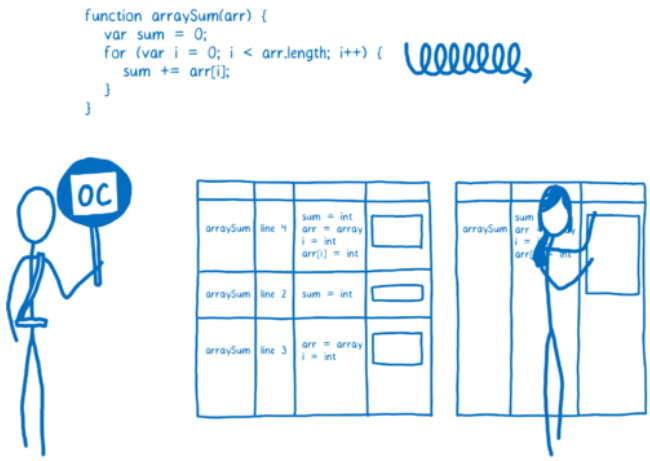
Source: https://hacks.mozilla.org/2017/02/a-cartoon-intro-to-webassembly



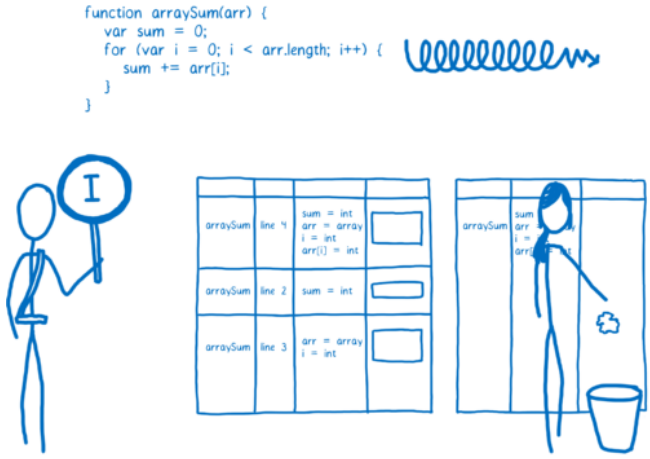
Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/



Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/



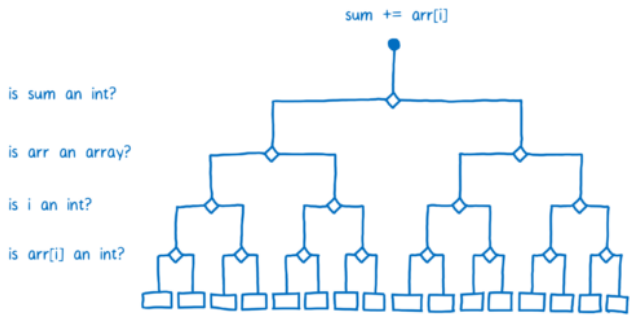
Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/



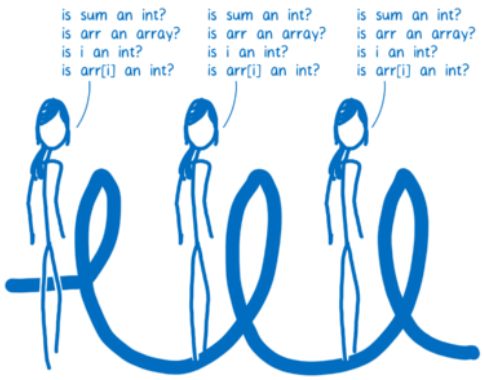
Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/

Sample code

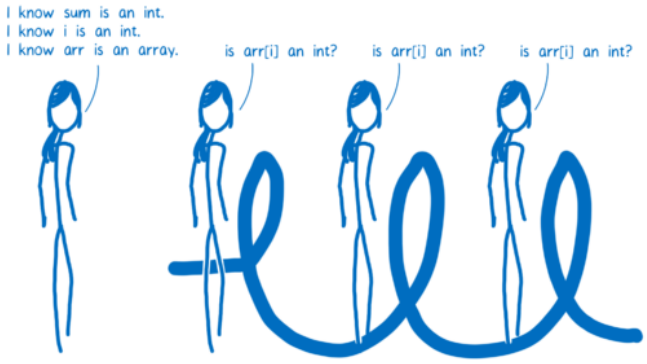
```
function arraySum(arr) {
  var sum = 0;
  for (var i = 0; i < arr.length; i++) {
    sum += arr[i];
  }
}
```

Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/



Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/



Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/

https://lists.w3.org/Archives/Public/public-webassembly/2017Feb/0002.html

"WebAssembly or wasm is a new portable, size- and load-time-efficient format suitable for compilation to the web."

Source: http://webassembly.org

Qt Wiggly

Qt Animated Tiles

http://webassembly.org/demo/

Emscripten

What is it?

What is it?

- Open Source LLVM to Javascript compiler

What is it?

- Open Source LLVM to Javascript compiler
- Compile C/C++ into Javascript

What is it?

- Open Source LLVM to Javascript compiler
- Compile C/C++ into Javascript
- Convert any LLVM bitcode output into Javascript

What is it?

- Open Source LLVM to Javascript compiler
- Compile C/C++ into Javascript
- Convert any LLVM bitcode output into Javascript
- Compile C/C++-based runtimes into Javascript (Python/Lua)

Why is it?

Why is it?

- JavaScript engines are fast and getting faster

Why is it?

- JavaScript engines are fast and getting faster
- People are familiar with other tools and languages

Why is it?

- JavaScript engines are fast and getting faster
- People are familiar with other tools and languages
- JavaScript kind of sucks

Why is it?

- JavaScript engines are fast and getting faster
- People are familiar with other tools and languages
- JavaScript kind of sucks
- Code is compiled into LLVM bitcode which unlocks further optimizations

Compiling other languages to JS

"Emscripten makes native code immediately available on the Web: a platform that is standards-based, has numerous independent compatible implementations, and runs everywhere from PCs to iPads."

Source: http://kripken.github.io/emscripten-site/docs/introducing_emscripten/about_emscripten.html

"With Emscripten, C/C++ developers don’t have the high cost of porting code manually to JavaScript — or having to learn JavaScript at all. Web developers also benefit, as they can use the many thousands of pre-existing native utilities and libraries in their sites."

Source: http://kripken.github.io/emscripten-site/docs/introducing_emscripten/about_emscripten.html

Hello, World!

```c
#include <stdio.h>

int main() {
  printf("hello, world!\n");
  return 0;
}
```

Hello, World!

```c
#include <stdio.h>

int main() {
  printf("hello, world!\n");
  return 0;
}
```

```
$ ./emcc hello.c
```


Hello, World!

```c
#include <stdio.h>

int main() {
  printf("hello, world!\n");
  return 0;
}
```

```
$ ./emcc hello.c
```

a.out.js

Hello, World!

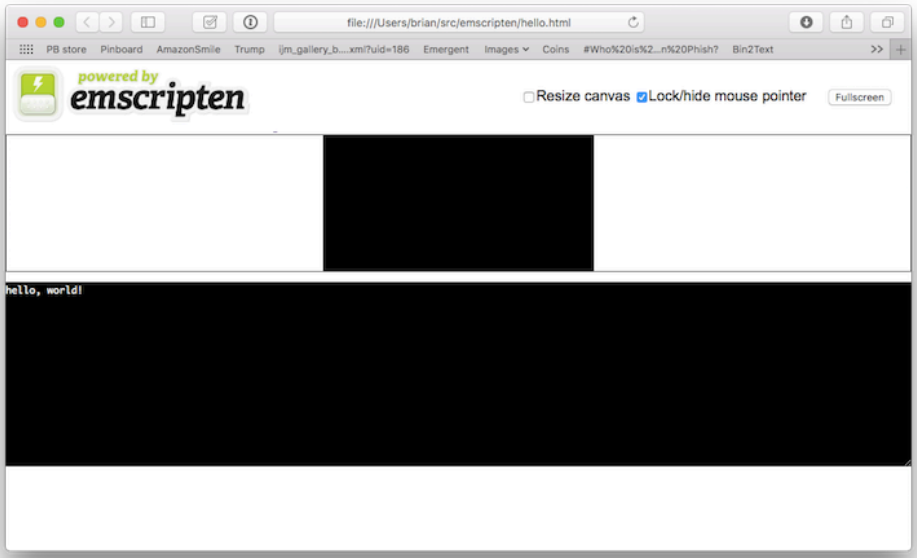
```c
#include <stdio.h>

int main() {
  printf("hello, world!\n");
  return 0;
}
```

```
$ ./emcc hello.c
```

a.out.js

```
$ node a.out.js
hello, world!
```


Any Modern Browser



File I/O

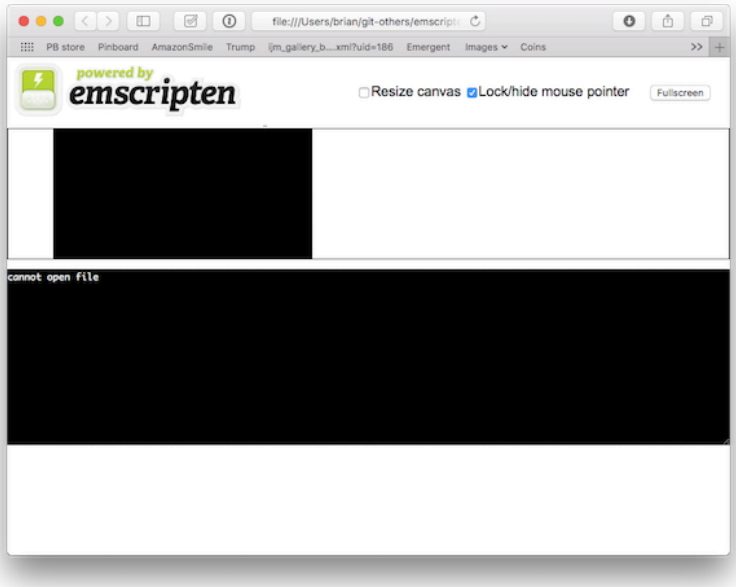
```c
#include <stdio.h>
int main() {
  FILE *file = fopen("tests/hello_world_file.txt", "rb");
  if (!file) {
    printf("cannot open file\n");
    return 1;
  }
  while (!feof(file)) {
    char c = fgetc(file);
    if (c != EOF) {
      putchar(c);
    }
  }
  fclose (file);
  return 0;
}
```


File I/O

```c
#include <stdio.h>
int main() {
  FILE *file = fopen("tests/hello_world_file.txt", "rb");
  if (!file) {
    printf("cannot open file\n");
    return 1;
  }
  while (!feof(file)) {
    char c = fgetc(file);
    if (c != EOF) {
      putchar(c);
    }
  }
  fclose (file);
  return 0;
}
```

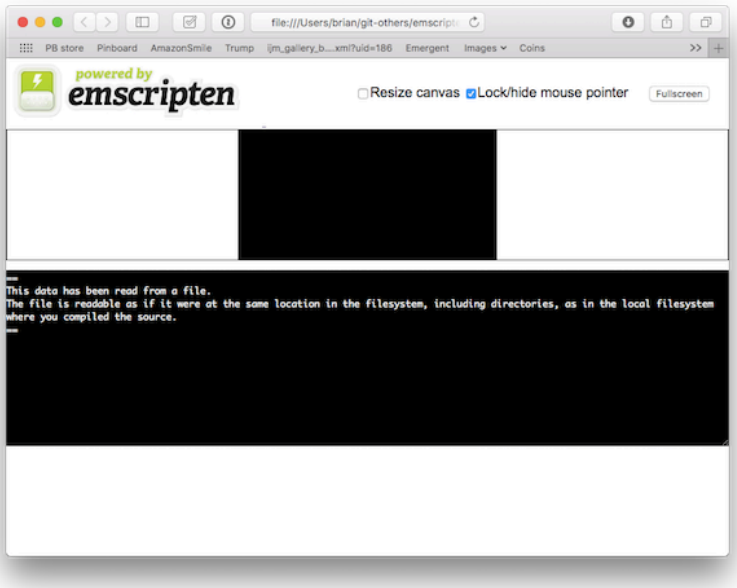
```
$ ./emcc tests/hello_world_file.cpp -o hello.html
```

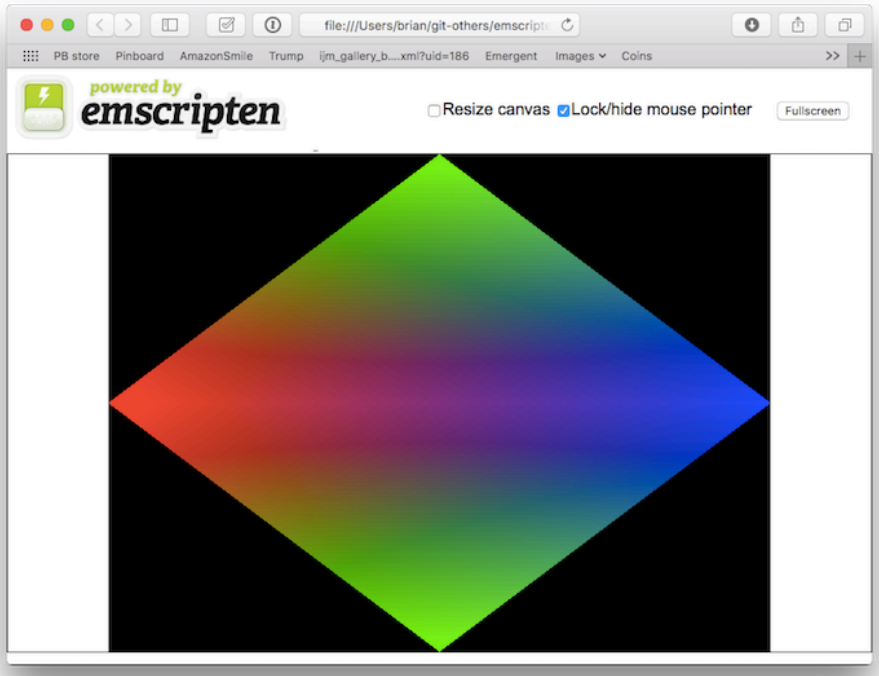
No Preload of Files




```
$ ./emcc tests/hello_world_file.cpp -o hello.html --preload-file tests/hello_world
```

```
$ ./emcc tests/hello_world_file.cpp -o hello.html --preload-file tests/hello_world
```





WebAssembly

Efficient and fast

- Stack machine

Efficient and fast

- Stack machine
- Size and load-time efficient binary format

Efficient and fast

- Stack machine
- Size and load-time efficient binary format
- Portable with near native execution speeds

Safe

- Sandboxed execution environment

Safe

- Sandboxed execution environment
- Enforce same-origin and browser security policies

Open

- Easy to read textual format

Open

- Easy to read textual format
- Supports debugging, testing, optimizations, experiments

Part of the Web

- Maintains versionless, feature-tested, backwards-compatibility of the Web

Part of the Web

- Maintains versionless, feature-tested, backwards-compatibility of the Web
- Modules can interact with Javascript context and browser functionality

Part of the Web

- Maintains versionless, feature-tested, backwards-compatibility of the Web
- Modules can interact with Javascript context and browser functionality
- Also supports non-web embeddings

Binary Format

Binary Format

- Gzipped source code compresses very nicely

Binary Format

- Gzipped source code compresses very nicely
- Experiments demonstrated a 20-30% size reduction of a binary format vs asm.js code

Binary Format

- Gzipped source code compresses very nicely
- Experiments demonstrated a 20-30% size reduction of a binary format vs asm.js code
- Parse times of names is slower than binary indices

Binary Format

- Gzipped source code compresses very nicely
- Experiments demonstrated a 20-30% size reduction of a binary format vs asm.js code
- Parse times of names is slower than binary indices
- Further optimizations are possible

Binary Layering

Binary Layering

- Layer 0: Simple binary encoding of bytecode and data structures

Binary Layering

- Layer 0: Simple binary encoding of bytecode and data structures
- Layer 1: Structural compression layer *

Binary Layering

- Layer 0: Simple binary encoding of bytecode and data structures
- Layer 1: Structural compression layer *
- Layer 2: General compression *

Binary Layering

- Layer 0: Simple binary encoding of bytecode and data structures
- Layer 1: Structural compression layer *
- Layer 2: General compression *

* Future feature

Text format

http://webassembly.org/docs/jit-library/

http://webassembly.org/docs/mvp/

http://webassembly.org/docs/use-cases/

Building Emscripten w/ WebAssembly Support

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

Building Emscripten w/ WebAssembly Support

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

```
$ source ./emsdk_env.sh
Adding directories to PATH:
PATH += /Users/brian/git-others/emsdk/emsdk
PATH += /Users/brian/git-others/emsdk/emsdk/clang/fastcomp/build_incoming_64/bin
PATH += /Users/brian/git-others/emsdk/emsdk/node/4.1.1_64bit/bin
PATH += /Users/brian/git-others/emsdk/emsdk/emscripten/incoming

Setting environment variables:
EMSDK = /Users/brian/git-others/emsdk/emsdk
EM_CONFIG = /Users/brian/.emscripten
EMSCRIPTEN = /Users/brian/git-others/emsdk/emsdk/emscripten/incoming
```


Testing the Build

```
$ emcc -v
WARNING:root:(Emscripten: system change: 1.37.3|/Users/brian/git-others/emsdk/emsd
INFO:root:(Emscripten: Running sanity checks)
emcc (Emscripten gcc/clang-like replacement + linker emulating GNU ld) 1.37.3
clang version 3.9.0 (https://github.com/kripken/emscripten-fastcomp-clang/ 5725e3d
Target: x86_64-apple-darwin16.4.0
Thread model: posix
InstalledDir: /Users/brian/git-others/emsdk/emsdk/clang/fastcomp/build_incoming_64
INFO:root:(Emscripten: Running sanity checks)
```

Hello, World!

```c
#include <stdio.h>

int main() {
  printf("hello, world!\n");
  return 0;
}
```


Compiling to WASM/HTML

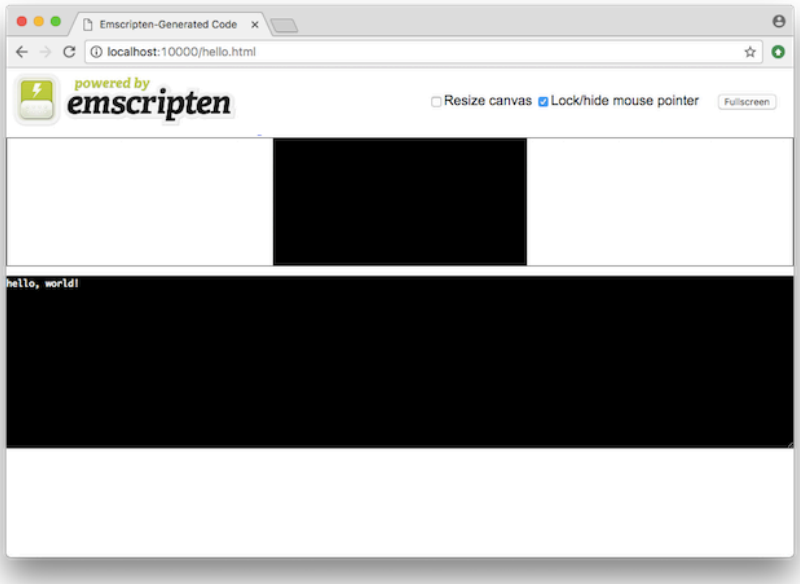
```
$ emcc hello.c -s WASM=1 -o hello.html
$ ls -alF
total 960
drwxr-xr-x   7 brian  staff     238 Feb 19 17:05 ./
drwxr-xr-x  15 brian  staff     510 Feb 19 17:05 ../
-rw-r--r--   1 brian  staff  235507 Feb 19 17:05 hello.asm.js
-rw-r--r--   1 brian  staff      76 Feb 19 17:05 hello.c
-rw-r--r--   1 brian  staff  103079 Feb 19 17:05 hello.html
-rw-r--r--   1 brian  staff   90634 Feb 19 17:05 hello.js
-rw-r--r--   1 brian  staff   47496 Feb 19 17:05 hello.wasm
```

Compiling to WASM/HTML

```
$ emcc hello.c -s WASM=1 -o hello.html
$ ls -alF
total 960
drwxr-xr-x   7 brian  staff     238 Feb 19 17:05 ./
drwxr-xr-x  15 brian  staff     510 Feb 19 17:05 ../
-rw-r--r--   1 brian  staff  235507 Feb 19 17:05 hello.asm.js
-rw-r--r--   1 brian  staff      76 Feb 19 17:05 hello.c
-rw-r--r--   1 brian  staff  103079 Feb 19 17:05 hello.html
-rw-r--r--   1 brian  staff   90634 Feb 19 17:05 hello.js
-rw-r--r--   1 brian  staff   47496 Feb 19 17:05 hello.wasm
```

```
$ emrun --no_browser --port 10000 .
Web server root directory: /Users/brian/src/emscripten/wasm
Now listening at http://localhost:10000/
The html page you are running is not emrun-capable. Stdout, stderr and exit(return
```


WebAssembly-Enabled Browser



Future

https://hacks.mozilla.org/2017/02/where-is-webassembly-now-and-whats-next/

http://webassembly.org/docs/gc/