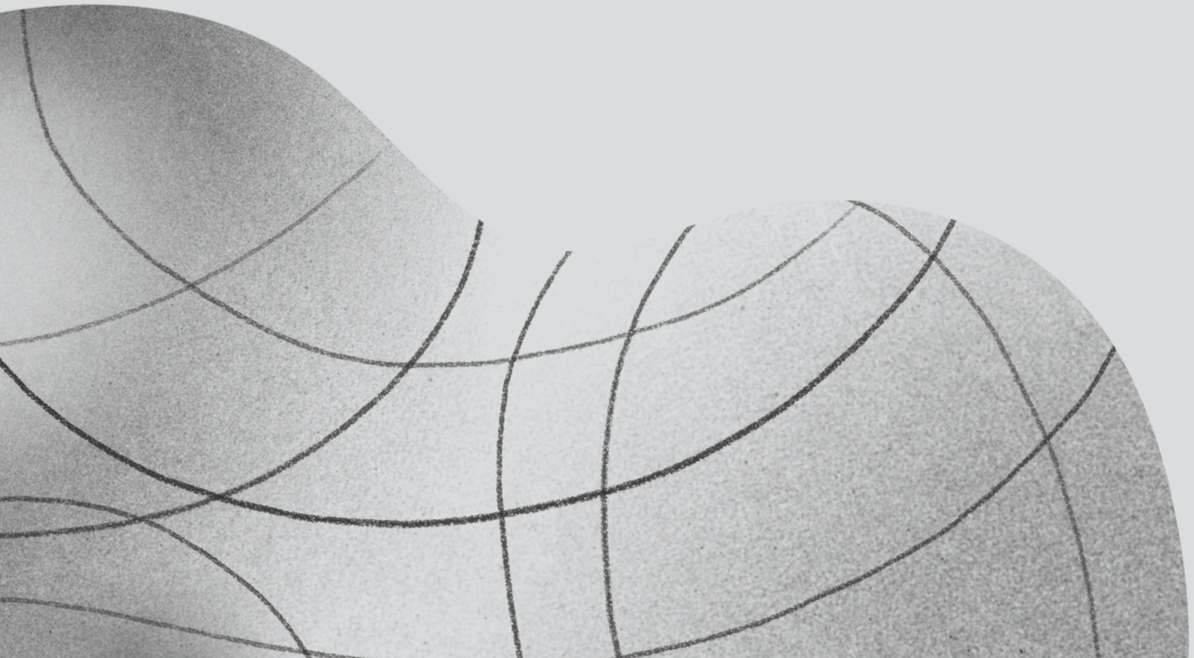
WEBASSEMBLY DEBUGGING

22. JUNI 2022
9:00 Uhr

Florian Schopp

Full Stack Developer
Florian_schopp@hotmail.com



WebAssembly is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.

Browser support:

Applications:

- Google Earth
- Unity
- AutoCAD

https://github.com/emscripten-core/emscripten/wiki/Porting-Examples-and-Demos

OUR MOTIVATION

- To use the same code base as the firmware.
- Include parts of the firmware libraries via WebAssembly
- Visualization of the effects of user changes on the signal

Home
Gerät
Verstärker
Auto-Null
Filter
Anwendungsmodus
Füller
Feldbus
Parametersätze
Gerätespeicher

Filter

Stufe 1

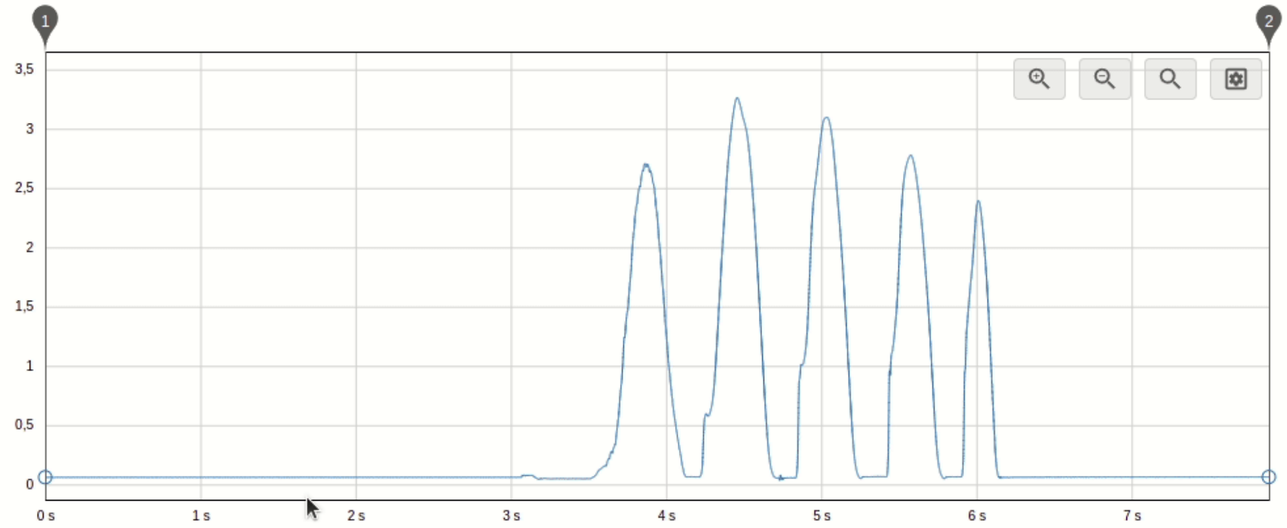
Filtertyp: -

Stufe 2

Filtertyp: -

ÄNDERUNGEN ANWENDEN

ZURÜCKSETZEN

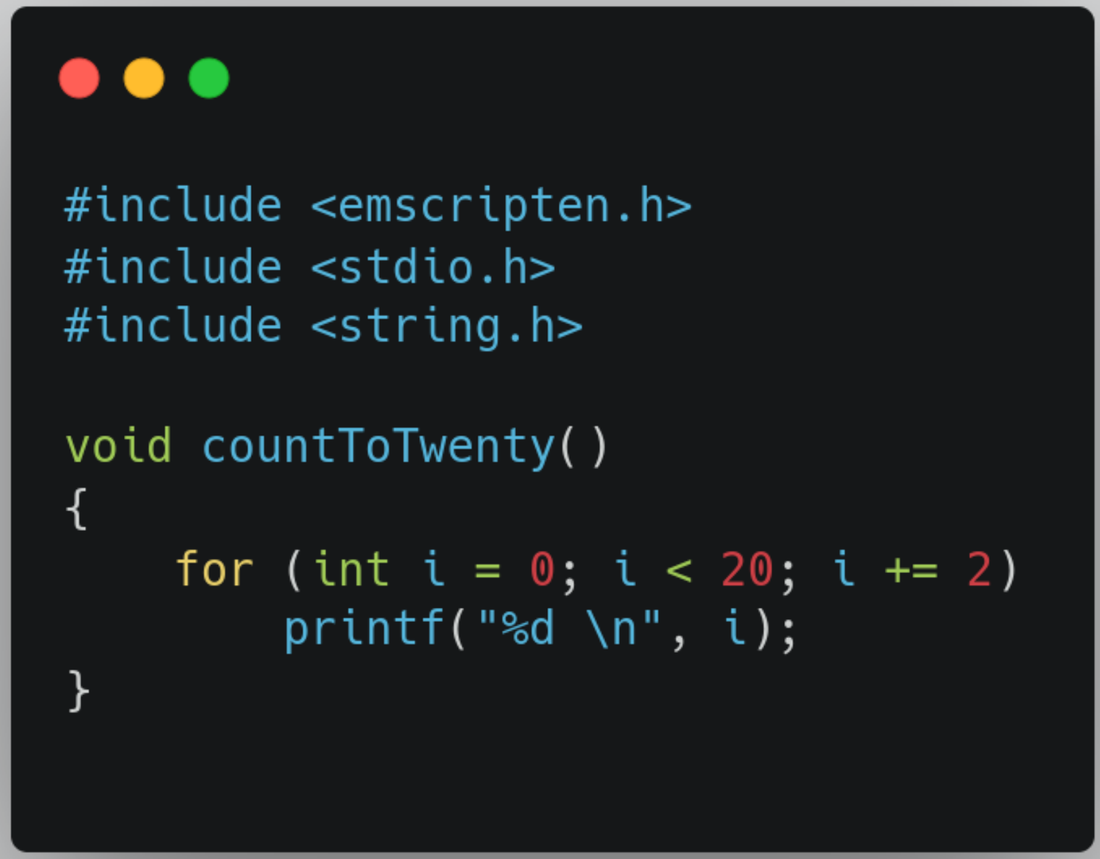


	1	2	Delta
Zeit	0,000 s	7,885 s	7,885 s /0,1 Hz
Ungefiltert	0,06 g	0,06 g	0,00 g
Geräteeinstellung	-	-	-
Simulation	-	-	-

AUFNAHME STARTEN

CSV-DATEI HERUNTERLADEN

WEBASSEMBLY GETTING STARTED

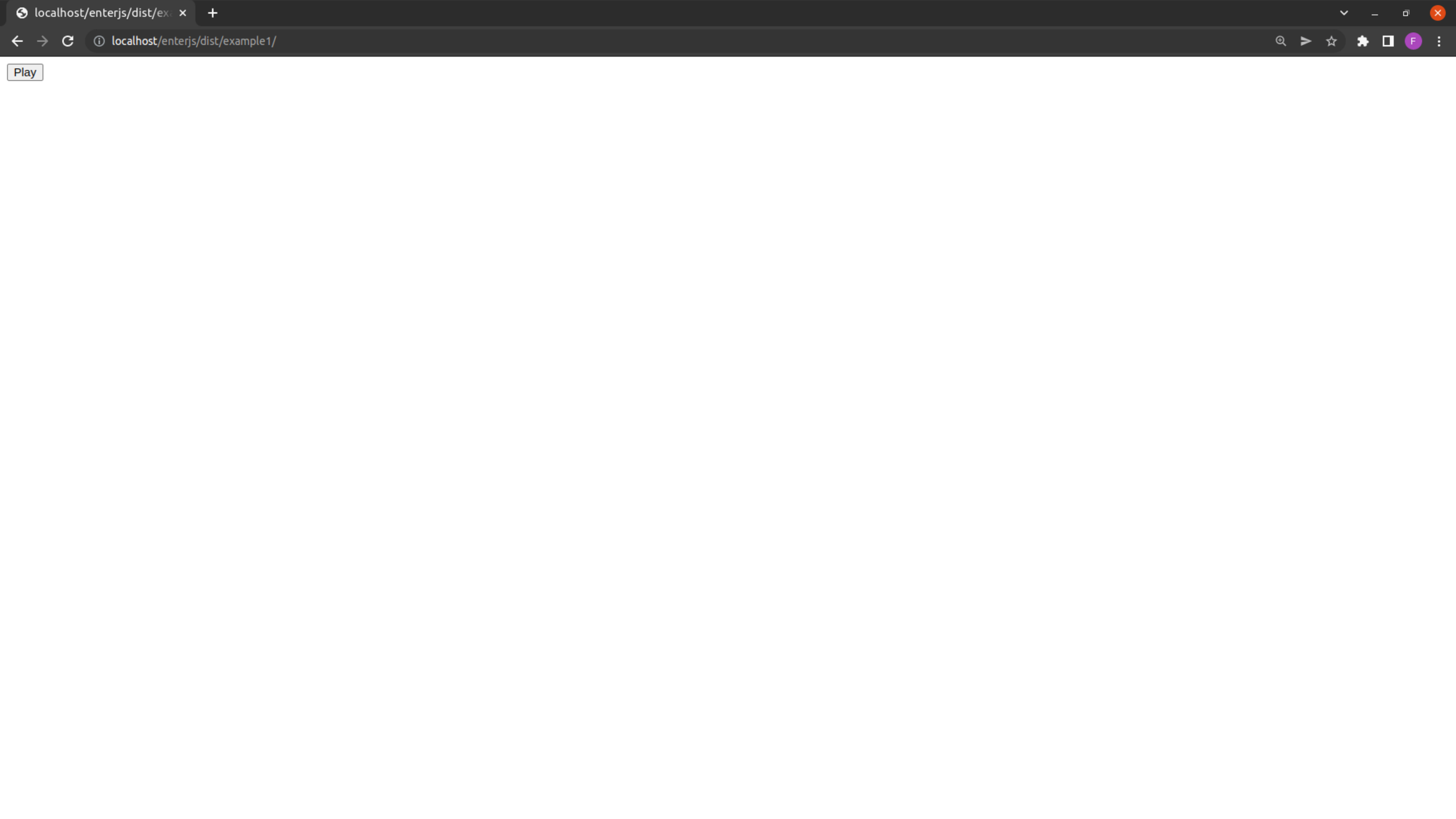


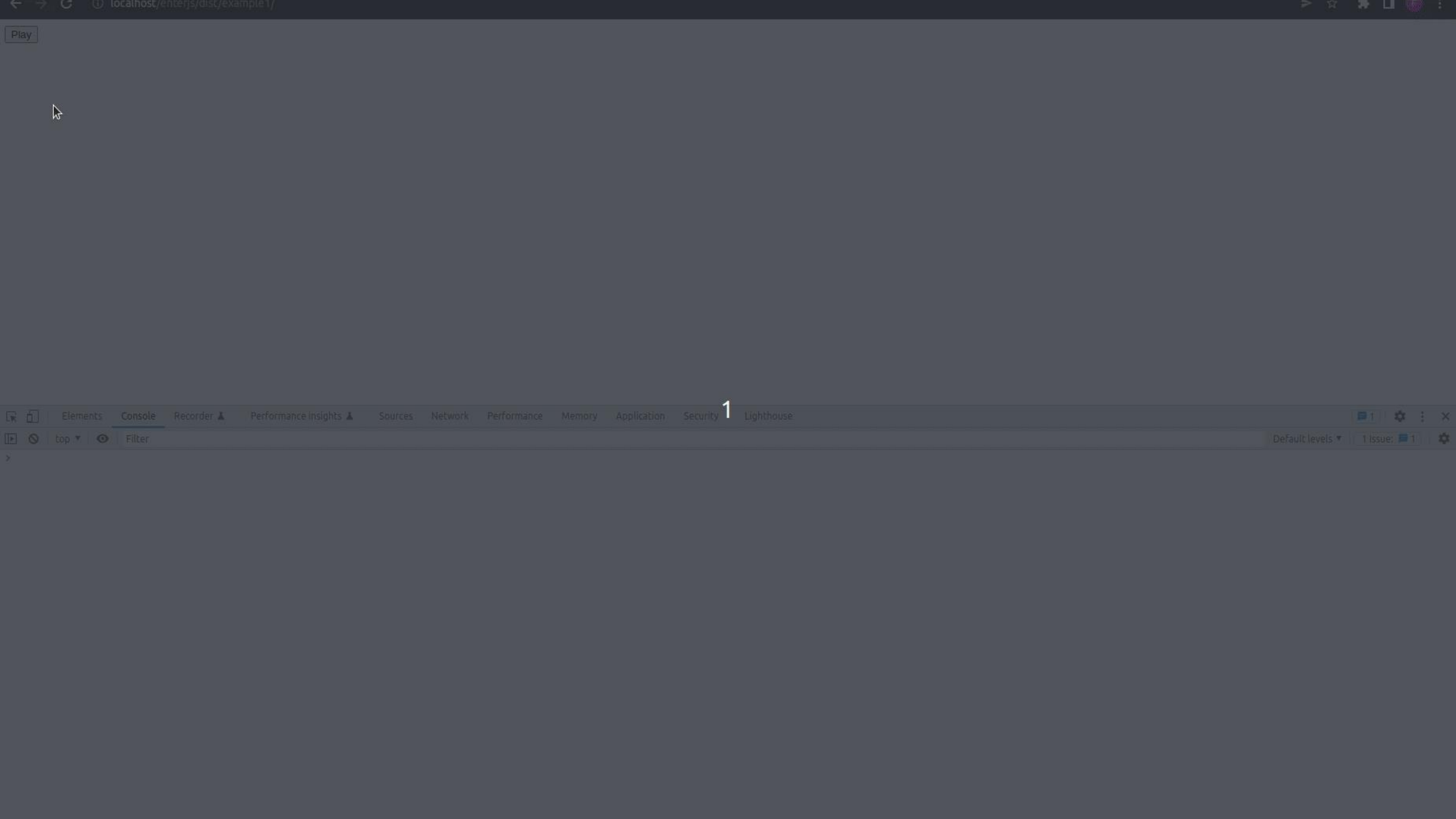
```c
#include <emscripten.h>
#include <stdio.h>
#include <string.h>

void countToTwenty()
{
    for (int i = 0; i < 20; i += 2)
        printf("%d \n", i);
}
```

emcc example2.c --no-entry -s EXPORT_ES6=1 -s MODULARIZE=1 -s EXPORTED_FUNCTIONS=_countToTwenty -o example1.js

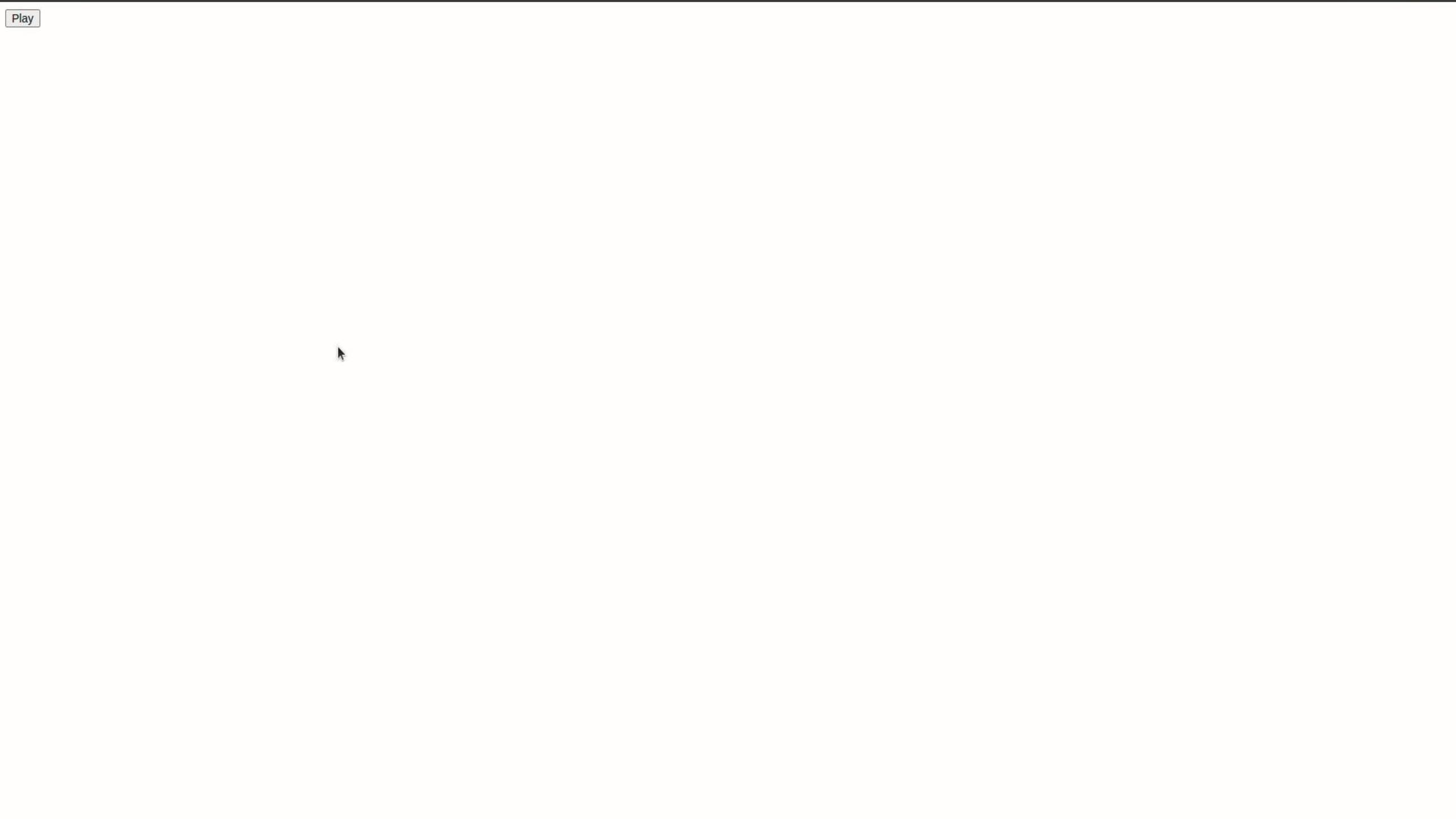

```
import initModule from './example1.js'
var loadedModule;
function playGame(){
    loadedModule._countToTwenty()

}
window.playGame = playGame
initModule().then(module=>{
    loadedModule=module

})
```

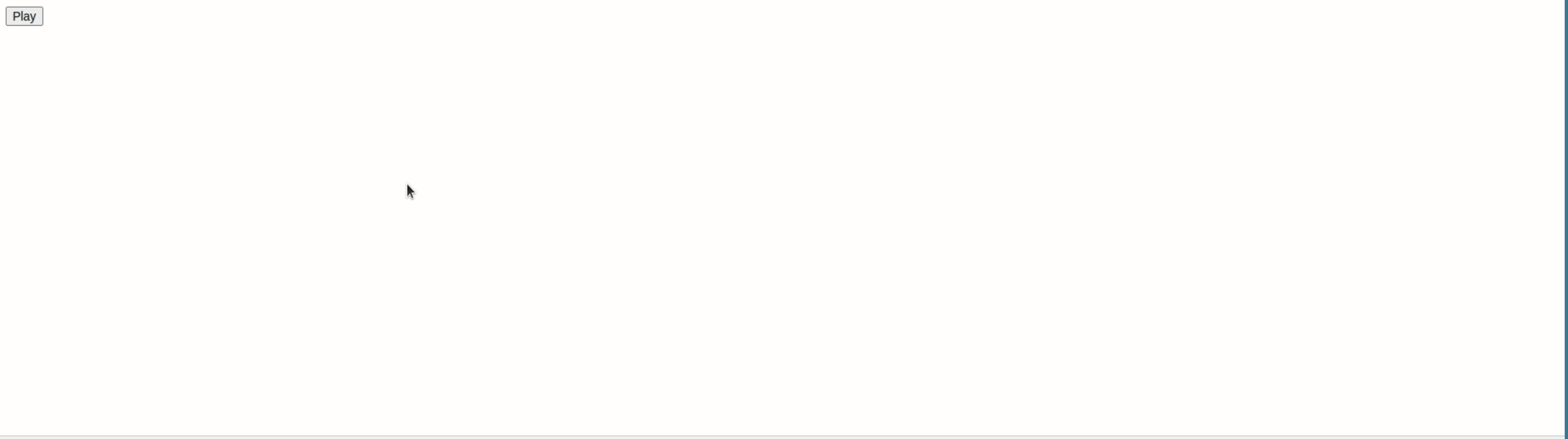
```
<html>
<body>
    <button onclick="playGame()">Play</button>
</body>
</html>
```


localhost/enterjs/dist/ex
localhost/enterjs/dist/example1/
Play


localhost/enterjs/dist/example1/
Play
1
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
Application
Security
Lighthouse
top
Filter
Default levels
1 Issue:

WEBASSEMBLY DEBUGGING


Play


Play
Elements
Console
Recorder
Performance insights
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Page
Filesystem
top
localhost
enterjs/dist/example1
(index)
calc.js
calc.wasm
(index)
calc.js
calc.wasm
<head>
<script type="module">
import initModule from './calc.js'
var loadedModule;
function playGame(){
loadedModule._countToTwenty()
}
window.playGame = playGame
initModule().then(module=>{
loadedModule=module
})
</script>
</head>
<body>
<button onclick="playGame()">Play</button>
</body>
Watch
Breakpoints
No breakpoints
Scope
Not paused
Call Stack
Not paused
XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints
CSP Violation Breakpoints
Coverage: n/a

DWARF SUPPORT IN CHROME

Activation

Debug-Console > Settings > Experiments > Enable DWARF support

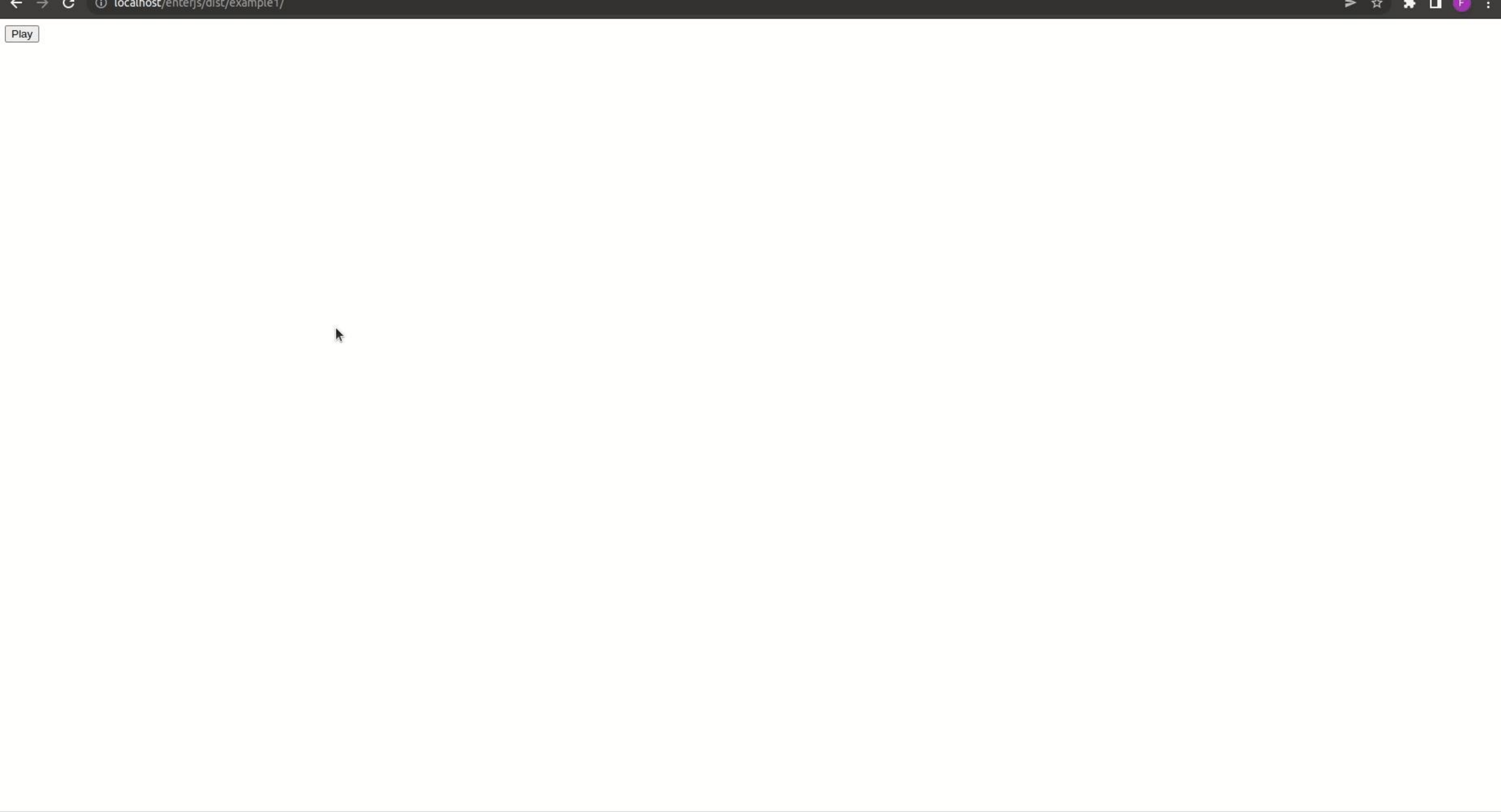
Install Extension

https://goo.gle/wasm-debugging-extension

Compile with debug information

-g


```
emcc example2.c --no-entry -s EXPORT_ES6=1 -s MODULARIZE=1 -s EXPORTED_FUNCTIONS=_countToTwenty -g -o example1.js
```



Enterjs game

Print Game Started

for 1 to 20
If the number is dividable by 3 and 5
print enterJS
If the number is dividable by 3 print enter
If the number is dividable by 5 print JS
Else print Cannot be devided by 3 or 5

Print Game Ended

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sanitizer/lsan_interface.h>

char *end, *str, *start;

char *checkNumber(int argument)
{
    if (!(argument % 3) & !(argument % 5))
        return "EnterJS";
    if (!(argument % 3))
        return "Enter";
    if (!(argument % 5))
        return "JS";
    return "Cannot be devided by three or five";
}

int enterjsGame()
{
    str = (char *)malloc(15);
    end = (char *)malloc(11);
    sprintf(end, "Game Ended");
    start = (char *)malloc(13);
    sprintf(start, "Game Started");

    printf("%s \n", start);
    for (int i = 0; i < 20; i++)
    {
        sprintf(str, "%s", checkNumber(i));
        printf("%d: %s \n", i, str);
    }
    printf("%s \n", end);
    return 0;
}
```


```
emcc example2.c --no-entry -s EXPORT_ES6=1 -s MODULARIZE=1 -s EXPORTED_FUNCTIONS=_enterjsGame -g -o example2.js
```

SANITIZERS

Purpose

Track the execution at runtime to report execution errors

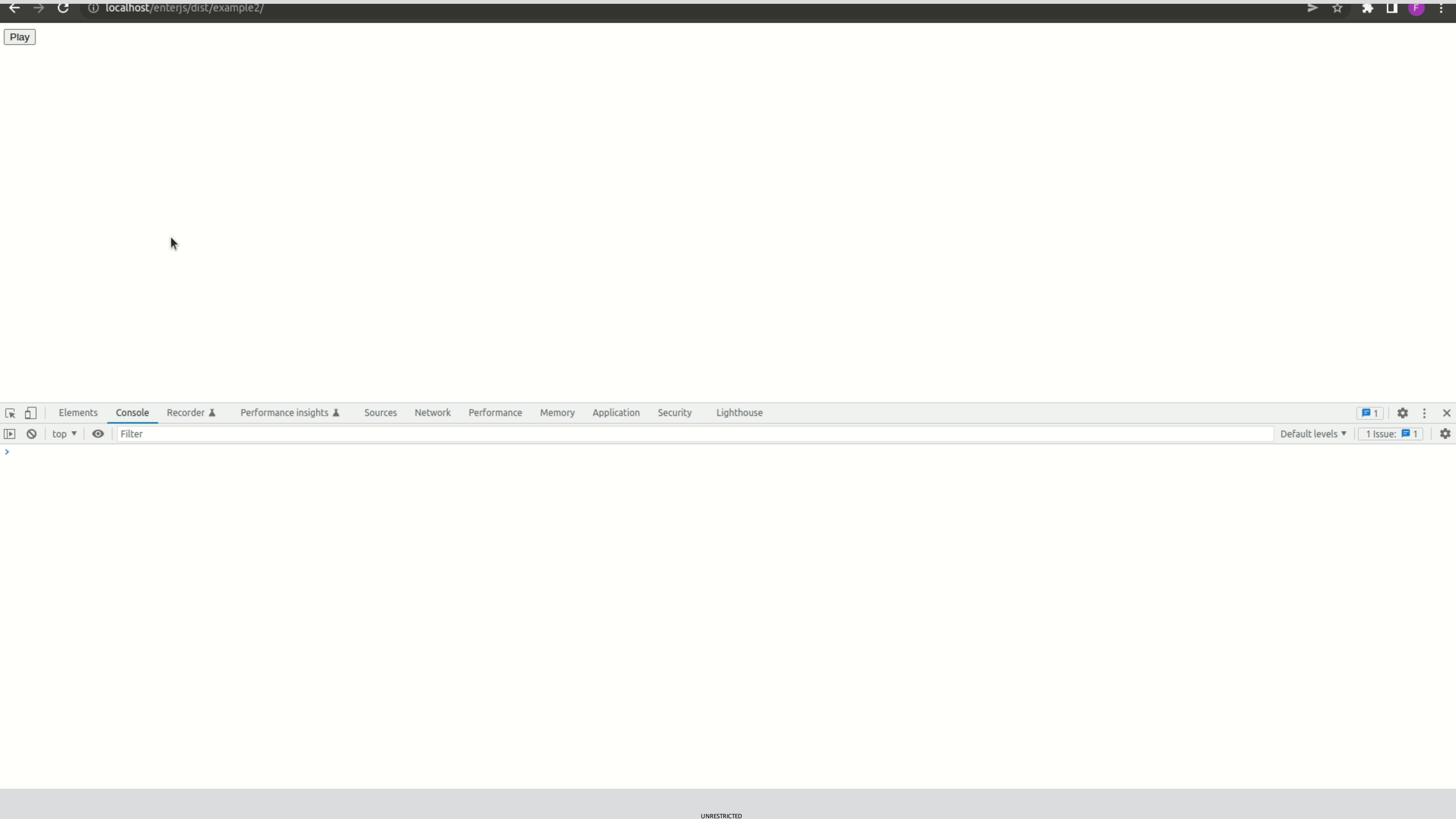
Types

- AddressSanitizer and LeakSanitizer
- ThreadSanitizer
- MemorySanitizer

Compile with debug information

– fsanitize=address


```
emcc example2.c --no-entry -s EXPORT_ES6=1 -s MODULARIZE=1 -s EXPORTED_FUNCTIONS=_enterjsGame -fsanitize=address -g -o example2.js
```



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sanitizer/lsan_interface.h>

char *end, *str, *start;

char *checkNumber(int argument)
{
    if (!(argument % 3) & !(argument % 5))
        return "EnterJS";
    if (!(argument % 3))
        return "Enter";
    if (!(argument % 5))
        return "JS";
    return "Cannot be devided by three or five";
}

int enterjsGame()
{
    str = (char *)malloc(15);
    end = (char *)malloc(11);
    sprintf(end, "Game Ended");
    start = (char *)malloc(13);
    sprintf(start, "Game Started");

    printf("%s \n", start);
    for (int i = 0; i < 20; i++)
    {
        sprintf(str, "%s", checkNumber(i));
        printf("%d: %s \n", i, str);
    }
    printf("%s \n", end);
    return 0;
}
```

Our problems

- Overlapping memory areas. Program used pointers that pointed into program stack.
- Memory was not freed after usage. Growing memory problem.
- Pointer to uninitialized memory areas

STANDALONE EXECUTION

Execution of Webassembly Modules outside of the browser

INSTALLATION

Wasm SDK

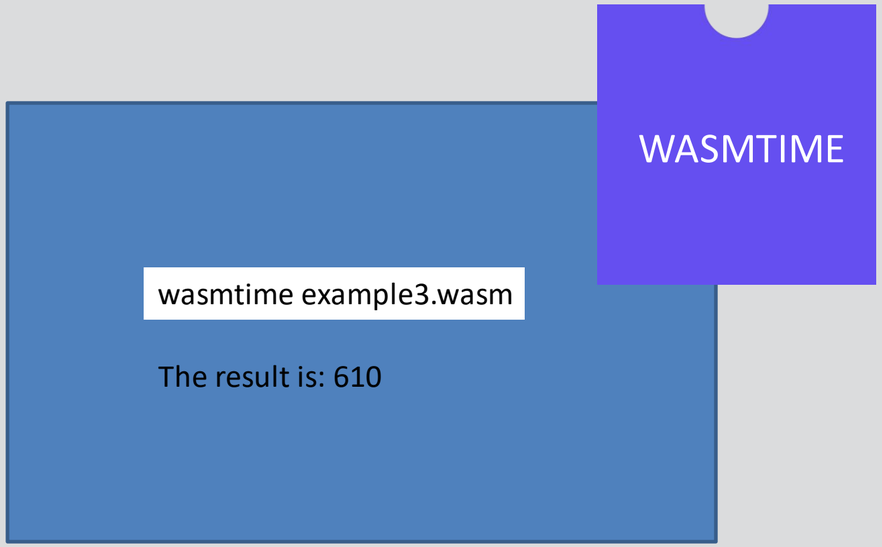
containing compatible clang package and sysroot
https://github.com/WebAssembly/wasi-sdk

Wasmtime

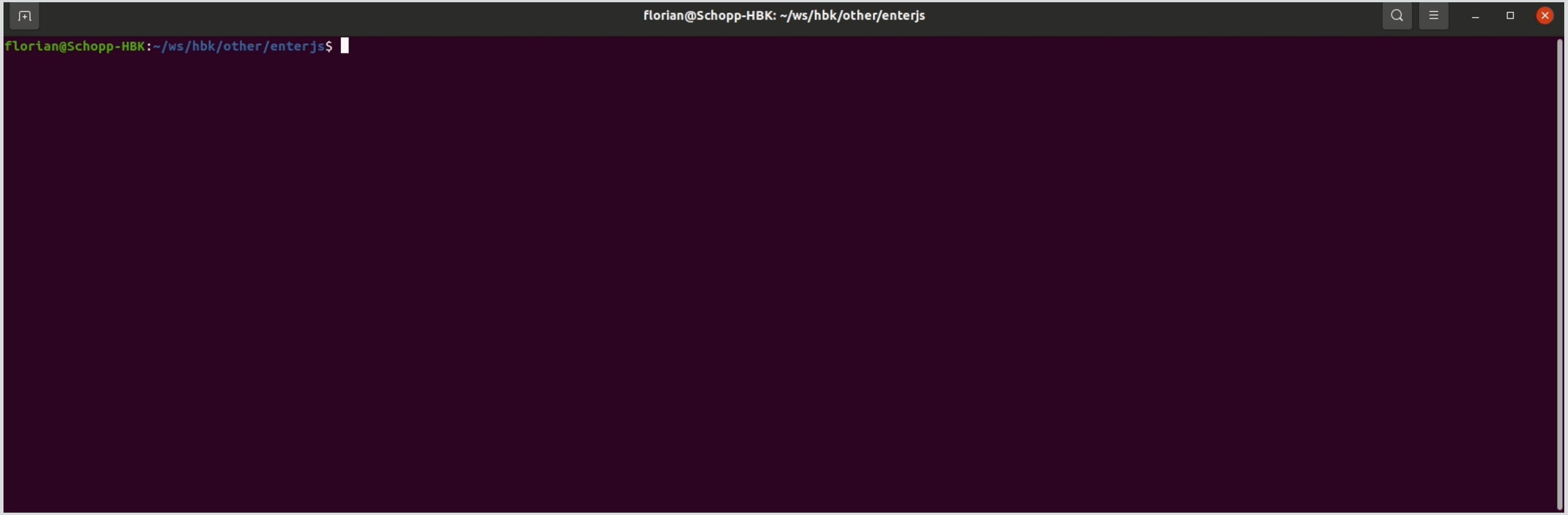
Standalone Runtime environment for webassembly
https://github.com/bytecodealliance/wasmtime

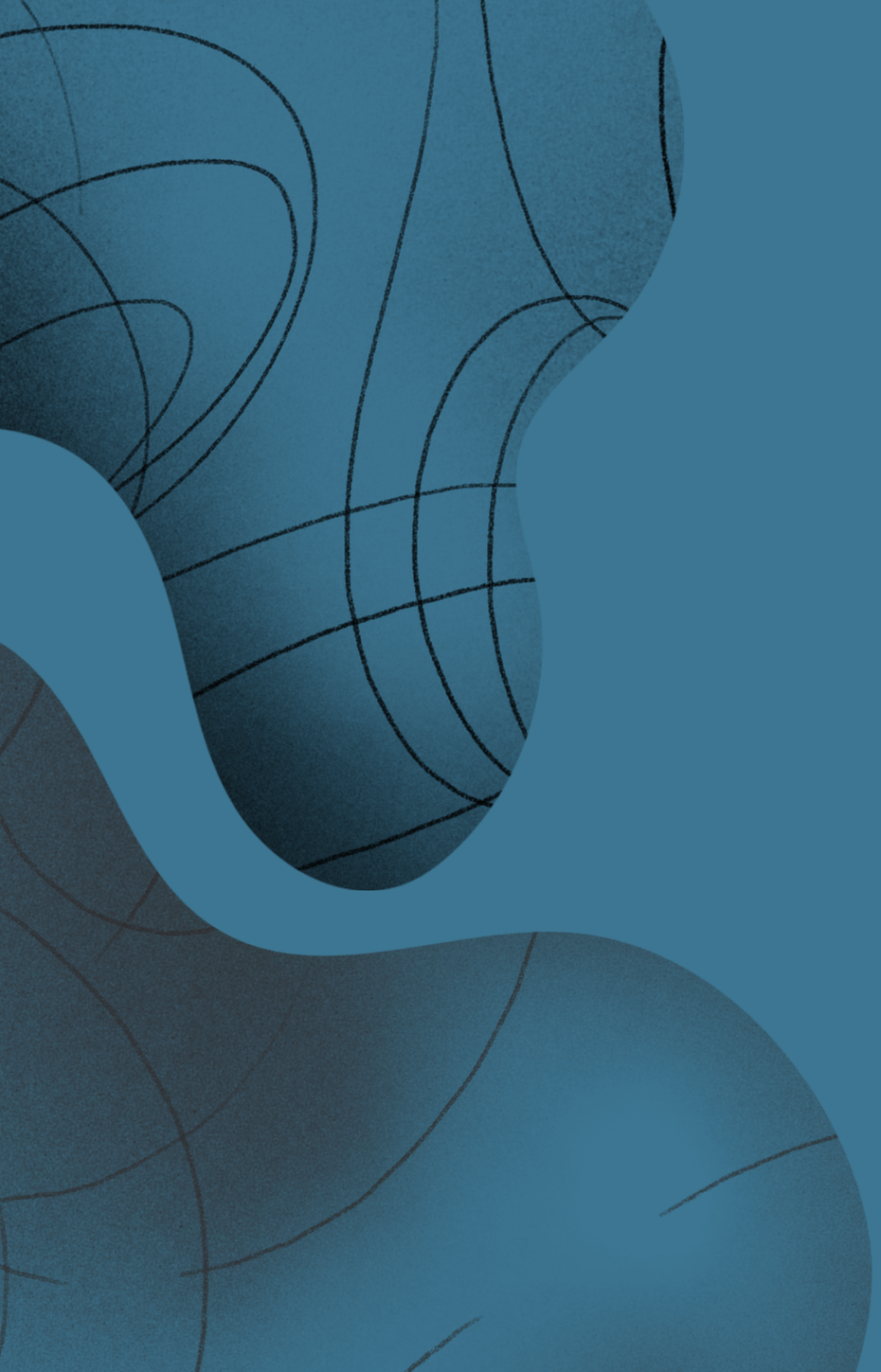
```c
#include <stdio.h>

int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
}
int main()
{
    int n = 15;
    printf("Result is: %d \n", fib(n));
    return 0;
}
```

clang-12 -- target=wasm32-wasi --sysroot=$WASI_SDK_PATH/share/wasi-sysroot - g example3.c - o example3.wasm



Attach to process using lldb





VS CODE INTEGRATION

```
{
    "tasks": [
        {
            "label": "compileWasm",
            "type": "cppbuild",
            "command": "/usr/bin/clang-12",
            "args": [
                "-fdiagnostics-color=always",
                "--target=wasm32-wasi",
                "--sysroot=$WASI_SDK_PATH/share/wasi-sysroot/",
                "-Xclang",
                "-disable-O0-optnone",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}/dist/${fileBasenameNoExtension}.wasm"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ],
    "version": "2.0.0"
}
```

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "lldb",
            "request": "launch",
            "name": "Launch",
            "program": "wasmtime",
            "args": ["run","${workspaceFolder}/${fileBasenameNoExtension}.wasm","-g"],
            "cwd": "${workspaceFolder}",
            "preLaunchTask": "compileWasm"
        }

    ]
}
```

RUN AND DEBUG Launch

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

- [x] C++: on throw
- [ ] C++: on catch

MODULES

example3.c

example3.c > fib(int)

```c
#include <stdio.h>

int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
}
int main()
{
    int n = 15;
    printf("Result is: %d \n", fib(n));
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Result is: 610
florian@Schopp-HBK:~/ws/hbk/other/enterjs$
```

bash
Launch
compileWasm

0 ⚠ 0 Launch (enterjs) Ln 8, Col 18 Spaces: 4 UTF-8 LF C Linux Prettier

THANK YOU

Github repository

https://github.com/Florian-Schopp/enterjs-2022