




# How 2 Webassembly

---

*Henry Rovnyak*

Go to TODO: PUT GITHUB LINK HERE and follow the setup  
instructions if you want to follow along



# What am I talking about?

---

- What is webassembly?
- How can I use it?
- But I'm a crazy person! How do I *\*really\** use it?

# What am I not talking about?

---

- Leptos
- Yew
- Blazor
- Vugu
- etc.

```
// Stolen from the Leptos book
#[component]
fn App() -> impl IntoView {
    let (count, set_count) =
        create_signal(0);

    view! {
        <button
            on:click=move |_| {
                set_count(3);
            }
        >
            "Click me: "
            {move || count()}
        </button>
    }
}
```

# How do websites work?

---

- HTML: Hypertext Markup Language
  - What to show
  - Requests resources
- CSS: Cascading Style Sheets
  - How the website should look
- Javascript
  - What the website should do

index.html:

```
<!doctype html>

<html>
  <head>
    <title>Mandelbrot</title>
  </head>

  <body>
    <canvas id="canvas" width="800"
height="600"></canvas>
    <script type="module" src="sketch.js"></
script>
  </body>
</html>
```

# What problems does Webassembly solve?

- Javascript is a bad language
  - Cursed
  - No type checking
  - Slow
- Running stuff on the web that was not meant for the web



```
"0" == 0 // true
0 == [] // true
"0" == [] // false???
```

# What is Webassembly?

---

- Similar to assembly/machine code
- Compilation target for other languages
- Small instruction set
- Interfaces with Javascript
- Needs Javascript glue code
- No access to the standard library

# Demo 1!

---

- Mandelbrot set
  - You should already have everything set up if you want to follow along

Now observe this Meme

---





I know what you're thinking...

I know what you're thinking...

W

A

T

And you'd be right!

Web

Assembly

Text

# Sections

---

- Globally accessible stuff
  - Memory
  - Globals
  - Imports
  - Exports
  - Functions
  - etc.

```
(module
  (memory (import "js" "mem") 1)
  (global $end_of_input (import "js"
    "endOfInput") i32)

  (import "console" "log" (func $log (param i32)
    (result i32)))
  (import "js" "parseNum" (func $parse_num (param
    i32) (result i32)))

  (func (export "solve") (result i32)
    ; ...
  )
)
```

# Instructions/Execution

---

- Small instruction set
- Mix of stack and register based
- Higher level control flow constructs

# Demo!

---

<https://adventofcode.com/2023/day/4>