SAN DIEGO RUST MEETUP

RUSTWASM

SEPTEMBER 13TH, 2018

- Presenter
  - Brian Manning
  - https://github.com/spicyjack
- San Diego Rust
  - https://gitter.im/sdrust/Lobby
  - https://github.com/sdrust
  - https://github.com/SDRust/rustwasm_notes

A LOW-LEVEL ASSEMBLY-LIKE LANGUAGE WITH A COMPACT BINARY FORMAT THAT RUNS WITH NEAR-NATIVE PERFORMANCE AND PROVIDES HIGH-LEVEL LANGUAGES WITH A COMPILATION TARGET SO THAT THEY CAN RUN ON THE WEB

What is WebAssembly (WASM)?

WHAT ARE THE BENEFITS OF WEBASSEMBLY?

- Efficient and fast
- Safe
- Open and debuggable
- Part of the “open web” platform

WHAT ARE THE LIMITATIONS OF WEBASSEMBLY?

- Very simple memory model
  - Only has access to a single “linear memory” (a flat array of a numeric type)
- Memory can only be grown in 64k pages
- Memory cannot be shrunk

EXAMPLE OF WEBASSEMBLY "TEXT" FORMAT

```
(module
  (func $fac (param f64) (result f64)
    get_local 0
    f64.const 1
    f64.lt
    if (result f64)
      f64.const 1
    else
      get_local 0
      get_local 0
      f64.const 1
      f64.sub
      call $fac
      f64.mul
    end)
  (export "fac" (func $fac)))
```

RUSTWASM = RUST + WEBASSEMBLY +

❤️

WHAT CAN YOU DO WITH RUSTWASM?

- From Rust, you can
  - Export to JS classes, functions, etc.
  - Manipulate the DOM
  - Log to the console
  - Do performance monitoring

EXAMPLES OF RUSTWASM

- Conway's Game of Life
  - https://github.com/rustwasm/wasm_game_of_life
    - Tutorial: https://rustwasm.github.io/book/introduction.html
- https://github.com/rustwasm/wasm-bindgen/tree/master/examples
  - Performance - https://github.com/rustwasm/wasm-bindgen/tree/master/examples/performance

WEBASSEMBLY & RUSTWASM LINKS

- WebAssembly
  - https://webassembly.org/
  - MDN
    - https://developer.mozilla.org/en-US/docs/WebAssembly
- RustWASM
  - https://github.com/rustwasm
  - Tutorial: https://rustwasm.github.io/book/introduction.html

DEMO

- Set up RustWASM for the Game of Life demo
  - `cargo install wasm-pack`
  - `cargo install cargo-generate`
  - You also need to have `npm` installed somewhere in your `$PATH`
  - Links to setup/workflow instructions
    - https://gitter.im/sdrust/Lobby

GROUP PROJECT IDEAS

- Beginner
  - When the page loads, show the grid, but don't automatically start calculating generations
  - Output universe calculation stats to _JavaScript_ `console.log`
    - Total calcuation time
    - Number of cells that changed state
- Intermediate/Advanced
  - Calculate sha256 sum in Rust from a string passed in from JavaScript
  - Initialize the universe with a single space ship.
  - Generate a random grid of cells (50% chance of alive or dead), instead of generating the same grid every time programatically
    - The grid generation is done in lib.rs (_Universe_ impl)
  - Insert a **glider** when the user Ctrl + Clicks on the grid
  - Insert a **pulsar** when the user Shift + Clicks on the grid