WEBASSEMBLY (WASM)

Demo

GOALS

GOALS

"Theory"

1. What is WASM?
2. Benefits of WASM in the browser

GOALS

"Theory"

1. What is WASM?
2. Benefits of WASM in the browser

Demo

1. Rust library (Collatz Conjecture)
2. Compile to WASM
3. Display Chart of Collatz Sequence

WHAT IS WASM?

- new type of code that can run in modern browsers
- low-level assembly-like language
- compact binary (runs with near-native performance)
- C/C++, C# and Rust can be compiled to WASM
- designed to run alongside JavaScript

BENEFITS OF WASM

1. 🏃‍♀️ PERFORMANCE
   - binary is fast
2. 🛠️ WE CAN USE RUST 💯
   - full control over memory usage (efficiency)
   - no garbage collection
   - no null pointer errors
   - safety

CREATE LIBRARY CRATE

```
# ❌ binary crate
#    src/main.rs
cargo new my_project

# ✅ library crate
#    src/lib.rs
cargo new --lib collatz_wasm
```

CREATE LIBRARY CRATE

```
# ❌ binary crate
#    src/main.rs
cargo new my_project

# ✅ library crate
#    src/lib.rs
cargo new --lib collatz_wasm
```

COLLATZ CONJECTURE (1)

```
// choose seed
let seed = ?;

// apply collatz algorithm
// (pseudocode)
if seed.is_even?
    return seed / 2
else // => seed is odd
    return 3 * seed + 1

// set new number as seed and repeat
// => until the sequence repeats
```

COLLATZ CONJECTURE (1)

```
// choose seed
let seed = ?;

// apply collatz algorithm
// (pseudocode)
if seed.is_even?
    return seed / 2
else // => seed is odd
    return 3 * seed + 1

// set new number as seed and repeat
// => until the sequence repeats
```

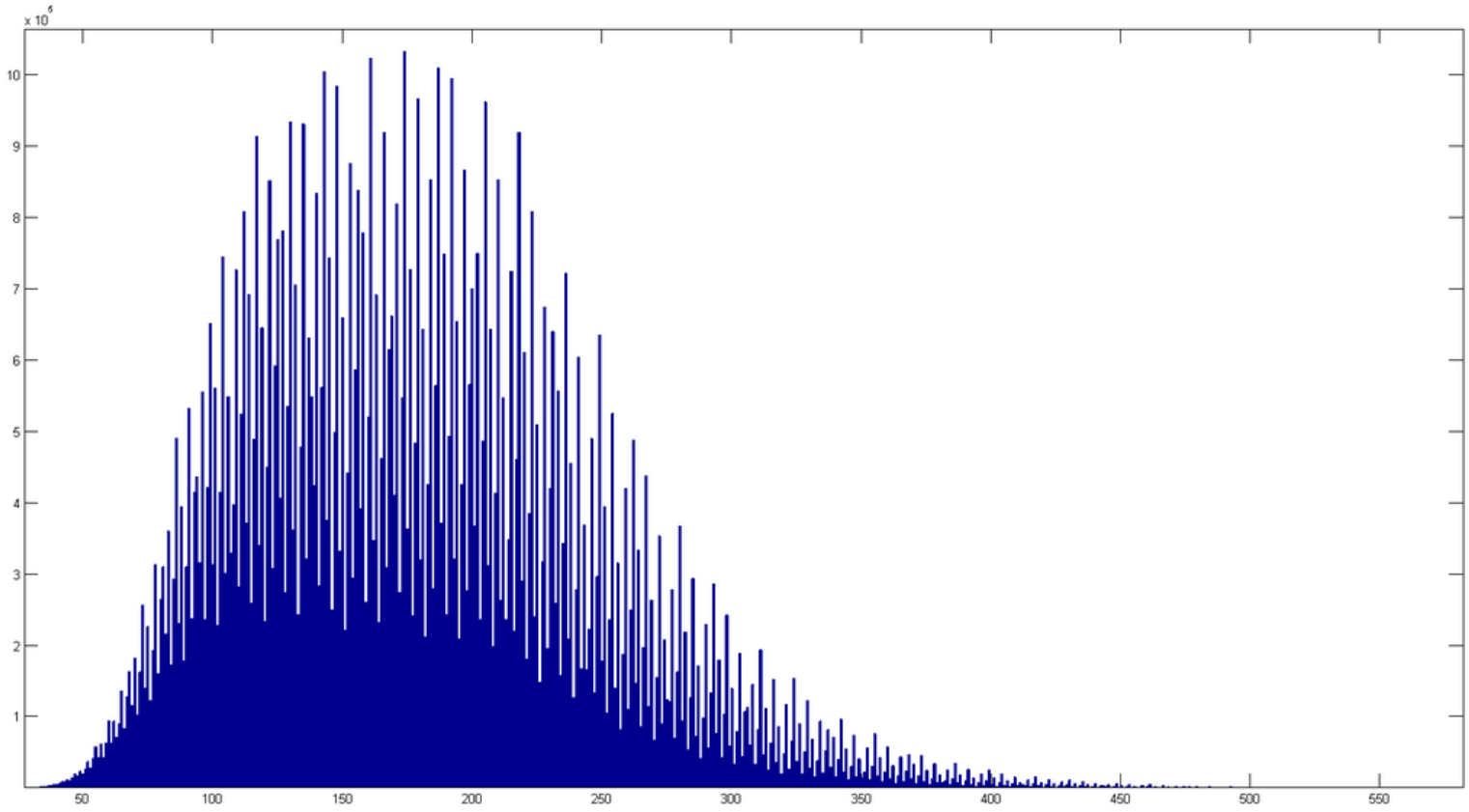
COLLATZ CONJECTURE (1)

```
// choose seed
let seed = ?;

// apply collatz algorithm
// (pseudocode)
if seed.is_even?
    return seed / 2
else // => seed is odd
    return 3 * seed + 1

// set new number as seed and repeat
// => until the sequence repeats
```

COLLATZ CONJECTURE (2)

```
collatz(4);  // => 4,2,1
collatz(5);  // => 5,16,8,4,2,1
collatz(6);  // => 6,3,10,5,16,8,4,2,1
collatz(7);  // => 7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1
collatz(8);  // => 8,4,2,1
collatz(10); // => 10,5,16,8,4,2,1
collatz(20); // => 20,10,5,16,8,4,2,1
```

COLLATZ CONJECTURE (3)



COLLATZ CONJECTURE IN RUST (1)

```rust
pub fn collatz_next_number(input: i32) -> i32 {
    if input % 2 == 0 { // if input is even
        input / 2
    } else { // if number is odd
        input * 3 + 1
    }
}
```

COLLATZ CONJECTURE IN RUST (1)

```rust
pub fn collatz_next_number(input: i32) -> i32 {
    if input % 2 == 0 { // if input is even
        input / 2
    } else { // if number is odd
        input * 3 + 1
    }
}
```

COLLATZ CONJECTURE IN RUST (1)

```rust
pub fn collatz_next_number(input: i32) -> i32 {
    if input % 2 == 0 { // if input is even
        input / 2
    } else { // if number is odd
        input * 3 + 1
    }
}
```

COLLATZ CONJECTURE IN RUST (2)

```rust
pub fn collatz(seed: i32) -> Vec<i32> {
    // initialize sequence with seed
    let mut sequence: Vec<i32> = vec![seed];
    let mut next_number: i32 = collatz_next_number(seed);
    while !sequence.contains(&next_number) {
        sequence.push(next_number);
        next_number = collatz_next_number(next_number);
    }
    sequence
}
```

COLLATZ CONJECTURE IN RUST (2)

```rust
pub fn collatz(seed: i32) -> Vec<i32> {
    // initialize sequence with seed
    let mut sequence: Vec<i32> = vec![seed];
    let mut next_number: i32 = collatz_next_number(seed);
    while !sequence.contains(&next_number) {
        sequence.push(next_number);
        next_number = collatz_next_number(next_number);
    }
    sequence
}
```

TEST OUR FUNCTION FIRST

```rust
#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn correct_sequences_generated() {
        assert_eq!(collatz(4), vec![/*4,2,1*/]);
        assert_eq!(collatz(5), vec![/*5,16,8,4,2,1*/]);
        assert_eq!(collatz(10), vec![/*10,5,16,8,4,2,1*/]);
    }
}
```

Run Tests

```
cargo test
```

TOOLING (1)

wasm-bindgen

```
# allows exporting Rust functionality to JS
# such as classes, functions, etc

# install by adding to Cargo.toml
[dependencies]
wasm-bindgen = "0.2.75"

# build
cargo build
```

TOOLING (1)

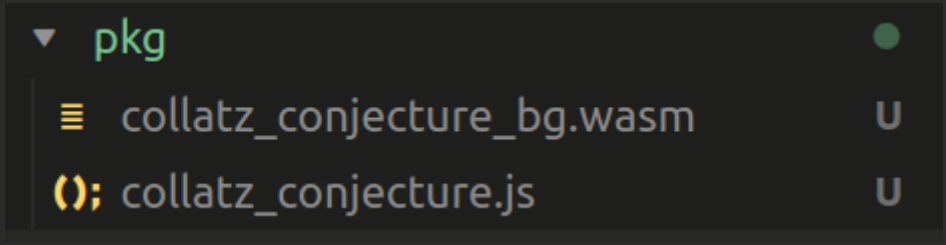
Update `src/lib.rs`

```rust
use wasm_bindgen::prelude::*;

#[wasm_bindgen]
fn collatz_next_number(input: i32) -> i32 {
    // [...]
}

#[wasm_bindgen]
fn collatz(seed: i32) -> Vec<i32> {
    // [...]
}
```

TOOLING (2)

wasm-pack

```
# CLI Tool to bundle rust binaries WASM javascript files

# install instructions
# => https://rustwasm.github.io/wasm-pack/installer/

# bundle your project
wasm-pack build --target web
```



CREATE MINIMAL WEBPROJECT

index.html

```html
<!-- import script -->
<script type="module" src="index.js"></script>
```

index.js

```javascript
import init, {
  collatz,
  collatz_next_number
} from "./pkg/collatz_wasm.js";

init().then(() => {
  const list = collatz(100);
  alert(list);
});
```

FINISHED PROJECT

index.html

index.js

styles.css

THE END

Thanks for your attention 🙏