Rust & Wasm

To build a browser based metronome

Vineet Naik Rust Bangalore meetup 20 July 2024

About me

- Using Rust for about 8-9 months
- Mostly a backend dev
 - Have done full stack dev prior to 2014
- No experience with audio programming
- Hobbyist guitar player

Outline

- What is Webassembly?
- Rust + Webassemby
 - Crates and tools for compiling rust to wasm
 - How to get rust code working inside the browser?
- What is a Metronome?
 - Demo
- Brief intro to WebAudio API
- Code examples
 - Debugging rust code at runtime
 - Error handling
 - Function callbacks and closures

Webassembly

What is Webassembly?

- Compact binary instruction
 format
- Not to be written by hand
 - Compilation target for Rust, C, C++
- Runs in modern web browsers
 - Designed to interop with Javascript both ways
 - But doesn't make any web specific assumptions
- Abbrev: Wasm in short



Made with 🏈 Whimsical

Advantages

- Near-native performance
 - Intermediate representation that's closer to machine code than JS source code
 - No garbage collection
- Compact binary format that's faster to transmit over network
 - Faster than even compressed JS
 - Tools available to further shrink the wasm binaries
- Can be used with existing JS code bases
 - \circ $\,$ No need to choose between Wasm and JS $\,$

Rust \rightarrow Wasm \rightarrow Browser

Two strategies

- 1. Build entire app in Rust
 - a. Yew framework
- 2. Wasm + JS
 - a. Build part of the app in Rust and integrate it into existing JS frontend

Tools

wasm-pack

- Tool to compile $Rust \rightarrow Wasm$
- Meant for interop with Javascript
 - \circ Either browser (web) or nodejs

\$ cargo install wasm-pack

wasm-bindgen

- A crate that wasm-pack depends on
- To provide a bridge between JS types and Rust types
- Comprises of multiple crates:
 - ∘ js_sys
 - bindings for std js objects only
 - web_sys
 - bindings for all the Web APIs that the browsers provide.
 - It's optional all APIs are gated by cargo features

How it works?



Made with 🏈 Whimsical

```
Cargo.toml
                                                                                             Cargo.lock
                                                                                             Cargo.toml
                                                                                             index.html
 1 [package]
                                                                                             pkg
                                                                                                hello_wasm.d.ts
 2 name = "hello-wasm"
                                        Ş
                                           wasm-pack build --target web
                                                                                                 hello wasm.js
 3 \text{ version} = "0.1.0"
                                                                                                 hello wasm bg.wasm
 4 edition = "2021"
                                                                                                hello_wasm_bg.wasm.d.ts
                                                                                                 package.json
                                                                                             src
6 [lib]
                                                                                                lib.rs
 7 crate-type = ["cdylib"]
                                                                                             target
                                                                                                CACHEDIR. TAG
                                                                                                debug
9 [dependencies]
                                                                                                release
10 wasm-bindgen = "0.2"
                                                                                                wasm32-unknown-unknown
                                                       🦲 🦲 🦲
                                                                                  index.html
                       lib.rs
                                                         1 <!doctype html>
                                                         2 <html lang="en-US">
 1 use wasm_bindgen::prelude::*;
                                                               <meta charset="utf-8" />
                                                               <title>hello-wasm example</title>
 3 #[wasm_bindgen]
                                                             </head>
 4 extern {
                                                             <body>
        pub fn alert(s: &str);
                                                               <script type="module">
 6 }
                                                                 import init, { greet } from "./pkg/hello_wasm.js";
                                                                 init().then(() => {
                                                                   greet("WebAssembly");
 8 #[wasm bindgen]
                                                                 });
 9 pub fn greet(name: &str) {
                                                               </script>
        alert(&format!("Hello, {}!", name));
                                                             </body>
11 }
                                                        15 </html>
```

Building a browser based metronome

Metronome

- Device that gives an audible click or beep at a set rate
- Used by musicians during practice
- Speed is measured in Beats Per Minute or BPM (also known as tempo)
 - 60 bpm = 1 beat per sec
 - 240 bpm = 4 beats per sec
 - 150 bpm = 2.5 beats per sec







Demo

https://naiquevin.github.io/rustick/

WebAudio API

- Browser API for controlling audio
- AudioContext
 - currentTime
- AudioNode
 - Source nodes
 - Effect nodes
 - Destination nodes
- AudioParam



Made with 🔇 Whimsical

Code structure

🔴 🔴 🔵 main.js	
<pre>1 const m = new Metronome() 2 .set_tempo(120) 3 .set_time_signature(4, 4</pre>	4)
<pre>4 .set_on_ended(() => { 5 console.log('A sound 6 }); 7</pre>	was played')
<pre>8 const timer = setInterval(9 () => m.schedule(), 10 m.beat_interval</pre>	
<pre>11); 12 13 // Stop 14 // clearInterval(timer):</pre>	

. lib.rs 1 use wasm_bindgen::prelude::*; 3 #[wasm_bindgen] 4 pub struct Metronome { 6 } 8 #[wasm_bindgen] 9 impl Metronome { #[wasm_bindgen(constructor)] pub fn new() -> Self { pub fn set_tempo(mut self, bpm: u16) -> Self { pub fn schedule(&mut self) -> {

WebAudio code in Rust

•••

Cargo.toml

- 1 [dependencies.web-sys]
- 2 version = "0.3.69"
- 3 features = [
- 4 "AudioContext",
- 5 "AudioDestinationNode",
- 6 "AudioParam",
- 7 "GainNode",
- 3 "OscillatorNode"
- 9]

2 let osc = self.ctx.create oscillator.unwrap(); 3 osc.frequency().set_value(800.0); // hertz 6 let envelope = self.ctx.create_gain().unwrap(); 7 envelope.gain().set value(1.0); 10 let play_time = self.scheduled_upto + self.beat interval; .gain() .exponential_ramp_to_value_at_time(1.0, play_time + 0.001) .unwrap(); .gain() .exponential_ramp_to_value_at_time(0.001, play_time + 0.02) .unwrap(); 23 osc.connect_with_audio_node(&envelope).unwrap(); .connect with audio node(&self.ctx.destination()) .unwrap(); 31 osc.start_with_when(play_time + 0.0).unwrap(); 34 osc.stop_with_when(play_time + 0.03).unwrap(); 37 self.scheduled_upto = play_time;

lib.rs

.

What if rust code panics?

- Panic converted to RuntimeError exception
- The stacktrace does point to text representation of wasm
- But symbols such as \$func86 are hardly of any help

Aside: Wasm text representation is a Lisp!

Uncaught RuntimeErr schedule timer setInterval hand runWasm runWasm async*	<pre>or: unreachable executed</pre>
<pre><anonymous> <anonymous> <anonymous> <anonymous> <anonymous> schedule timer (Async: setInter start</anonymous></anonymous></anonymous></anonymous></anonymous></pre>	<pre>http://localhost:8000/pkg/rustick_bg.wasm:15213 http://localhost:8000/pkg/rustick_bg.wasm:18031 http://localhost:8000/pkg/rustick_bg.wasm:20773 http://localhost:8000/pkg/rustick_bg.wasm:20773 http://localhost:8000/pkg/rustick_js:209 http://localhost:8000/metronome.js:84 val handler) http://localhost:8000/metronome.js:84</pre>
runWasm (Async: EventLis runWasm AsyncFunctionNex (Async: async) <anonymous></anonymous>	<pre>http://localhost:8000/main.js:32 tener.handleEvent) http://localhost:8000/main.js:31 t self-hosted:804 http://localhost:8000/main.js:37</pre>
00002057 unread	hable
00002058 end \$lab	
	el1
00002059 i32.cons	ell t 1
00002059 i32.cons 00002058 i32.cons	eti t 1 t 23
00002059 i32.cons 00002058 i32.cons 00002058 i32.cons 00002050 call \$fu	eli t 1 t 23 nc95
00002059 i32.com 00002059 i32.com 0000205B i32.com 0000205D call \$fu 0000205F unreachal	eli t 1 t 23 nc95 ble
00002059 i32.cons 00002058 i32.cons 00002050 call \$fu 0000205F unreachal 00002056 end \$label	el1 t 1 t 23 nc95 ble 3
00002055 i32.cons 00002055 i32.cons 00002055 i32.cons 00002055 call \$fu 00002057 unreachal 00002056 end \$label: 00002056 i32.cons*	el1 t 1 t 23 nc95 ble 3 1648652
00002055 cital state 00002055 cital state 00002055 cital state 00002055 call \$fu 00002055 unreachal 00002066 end \$label 00002061 i32.const 00002066 call \$fund	el1 t 1 t 23 nc95 ble 3 1048652 86
00002055 132.cons 00002055 132.cons 00002055 call \$fu 00002055 unreachal 00002056 end \$label 00002056 call \$furd 00002060 end \$label 00002061 132.const 00002066 call \$fund	e11 t 1 t 23 nc95 ble 3 1048652 86 e
00002055 i32.cons: 00002055 i32.cons: 00002055 call \$fu 00002057 unreachal 00002056 end \$label 00002056 call \$furct 00002060 end \$label 00002060 call \$funct 00002060 call \$funct 00002060 call \$funct 00002060 unreachable 00002069 end \$label4	e11 t 1 t 23 nc95 ble 3 1048652 86 e
00002055 i32.cons: 00002055 i32.cons: 00002055 call \$fu 00002057 unreachal 00002056 edl \$label 00002056 call \$fu 00002056 edl \$label 00002056 call \$fu 00002056 edl \$label 00002056 unreachal 00002068 unreachabl 00002069 edd \$label4 0000206A i32.const 1	el1 t 1 t 23 nc95 ble 3 10446552 86 e
00002055 i32.cons: 00002055 i32.cons: 00002055 call \$fu 00002057 unreachal 00002057 unreachal 00002056 end \$label: 00002056 call \$funct 00002056 call \$funct 00002056 unreachal 00002056 call \$funct 00002066 call \$label4 00002067 i32.const 13 00002066 i32.const 13	el1 t 1 t 23 nc95 ble 3 1048652 86 e
00002055 132.cons: 00002055 132.cons: 00002055 call \$fu 00002057 unreachall 00002056 end \$label 00002066 call \$funct 00002066 unreachall 00002066 unreachall 00002066 unreachall 00002066 i32.const 1 00002066 i32.const 1 00002066 i32.const 1 00002066 i32.const 1 00002066 call \$funct 00002066 call \$funct 00002066 call \$funct 00002066 call \$funct	el1 t 1 t 23 nc95 ble 3 1048652 86 e
000022050 i32.cons: 000022050 call \$fu 000022050 call \$fu 000022057 unreachal 000022056 end \$label 000022056 call \$funct 000022056 call \$funct 000022056 call \$funct 000022056 unreachable 000022059 end \$label4 00002059 end \$label4 00002050 i32.const 1 000022052 call \$functs 000022050 end \$label4 00002050 end \$label4 00002050 call \$functs 00002050 call \$functs 00002050 end \$label4	el1 t 1 t 23 nc95 ble 3 1048652 86 e
00002055 i32.cons: 00002055 i32.cons: 00002055 call \$fu 00002057 unreachal 00002056 end \$label 00002060 end \$label 00002066 call \$funct 00002066 unreachal 00002066 unreachal 00002069 end \$label4 00002069 end \$label4 00002060 call \$funct 00002060 unteachable 00002060 call \$funct 00002060 call \$not 1 00002060 call \$not 1 00002060 call \$funct 00002060 i32.const 1 00002060 call \$funct 00002060 ia2.const 1 00002060 unreachable 00002060 unreachable 00002060 unreachable 00002060 unreachable 00002070 unreachable	el1 t 1 t 23 nc95 ble 3 1044652 86 e
00002055 i32.cons* 00002055 i32.cons* 00002055 call \$fu 00002055 unreachal 00002056 edl \$label* 00002056 call \$funct 00002056 edl \$label* 00002060 end \$label* 00002060 i32.const 13 00002060 i32.const 13 00002060 call \$func95 00002070 unreachable 00002071) 00002072 (func \$func18	ell t 1 t 23 nc95 ble 3 1048652 86 e (param \$var0 i32) (param \$var1 i32) (param \$var2 i32) (result i32)
000022050 ci32.cons* 000022050 call \$fu 000022050 call \$fu 000022050 call \$fu 000022051 call \$fu 000022050 end \$label 000022051 i32.const 000022050 end \$label 000022050 i32.const 1 000022050 call \$functs 000022050 unreachable 000022070 unreachable 000022071) 000022072 (lucs \$functs	eli t 1 t 23 nc95 ble 3 1048652 86 e (param \$var0 i32) (param \$var1 i32) (param \$var2 i32) (result i32) (jaram \$var0 i32) (local \$var5 i32) (local \$var6 i32) (local
000022050 i32.const 000022050 call \$fu 000022050 call \$fu 000022057 unreachal 000022058 call \$fu 000022057 unreachal 000022050 call \$funcl 000022050 call \$funcl 000022050 call \$funcl 000022050 end \$label4 000022050 end \$label4 000022050 end \$label4 000022050 i32.const 1 000022050 call \$funcl5 000022050 end \$label4 000022050 end \$label4 000022050 i32.const 1 000022070 unreachable 000022071 (func \$funcl8 000022072 (local \$var3 000022077 block \$label4	ell t 1 t 23 nc95 ble 3 1048652 86 e (param \$var@ i32) (param \$var1 i32) (param \$var2 i32) (result i32) i32) (local \$var4 i32) (local \$var5 i32) (local \$var6 i32) (local 5

Build with --debug flag

\$ wasm-pack build --debug --target web

h32C hd49 hØd4 rust hc74 hb15 hb5b h43d metr sche time (Asy	6964d114e98b9 686a7fe3981a4 _begin_unwind 27f902a13f1a9 7b525de3fe68d acfb0dd292085 f5183222a4385 onome_schedule dule rr	<pre>http://localhost.8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_bg http://localhost:8000/pkg/rustick_js http://localhost:8000/pkg/rustick.js http://localhost:8000/pkg/rustick.js http://localhost:8000/pkg/rustick.js http://localhost:8000/metronome.js:8</pre>	.wasm:.03793 .wasm:63723 .wasm:83587 .wasm:77607 .wasm:79262 .wasm:77290 .wasm:83444 .wasm:2617 .wasm:49238 :258
star	t	http://localhost:8000/metronome.js:8	47
000C04B	local.get	\$var4	
000C04D 000C04F	local.get i32.add	\$var12	
000C050	local.set	\$var13	
000C052	local.get	\$var13	
000C054	local.get	\$var11	
000C056	call \$rust	ick::Metronome::schedule::h43df518	3222a4385
000C058	i32.const	36	
000C05A	local.set	\$var14	
000C05C	local.get	\$var4	
000C05E	local.get	\$var14	

00000060

i32 add

External crate: console_error_panic_hook

- Include the crate as a dependency
- Add following line in some code path that's guaranteed to execute e.g. Metronome::new function

```
panicked at src/lib.rs:144:28:
called `Option::unwrap()` on a `None` value
Stack:
__wbg_get_imports/imports.wbg.__wbg_new_abda76e88
logError@http://localhost:8000/pkg/rustick.js:88:
__wbg_get_imports/imports.wbg.__wbg_new_abda76e88
rustick.wasm.console error panic hook::Error::new
```

```
console_error_panic_hook::set_once()
;
```

Using console.log during development

```
#[wasm bindgen]
 4
 5
      extern "C" {
          // Use `js_namespace` here to bind `console.log(..)` instead of just
 6
7
          // `log(..)`
          #[wasm_bindgen(js_namespace = console)]
 8
9
          fn log(s: &str);
10
          // // Multiple arguments too!
11
          // #[wasm_bindgen(js_namespace = console, js_name = log)]
12
          // fn log specific(a: u8, b: f32);
13
      }
14
```

Or using helpers provided by the web-sys crate

- Web sys::console::log // array of values
- Web_sys::console::log 1 // single value
- Web sys::console::log 2 // two values



JsValue: Rust representation of an object owned by Javascript

Callback functions / Closures

```
• • •
                         lib.rs
                                                            .
                                                                                       example.rs
                                                              1 let closure: Closure<dyn Fn()> = Closure::new(move || {
  1 use web_sys::js_sys::Function;
                                                              3 });
  3 #[wasm bindgen]
  4 pub struct Metronome {
                                                              5 let window = web sys::window().expect('window is expected');
                                                              6 window.set interval with callback and timeout and arguments 0(
                                                                   closure.as_ref().unchecked_ref(),
        on ended: Option<Function>,
                                                                   1000.
                                                              9 ).unwrap();
 10 osc.set onended(self.on ended.as ref());
                                                             14 closure.forget();
 13 let f = self.on_ended.unwrap();
 14 let context = JsValue::null(); // bind to this
 15 f.call0(context).unwrap();
17 f.call1(context, JsValue::from bool(true));
 19 f.call2(context, _, _);
 20 f.call3(context, _, _, _);
```

References

Rustic metronome

- Code: <u>https://github.com/naiguevin/rustick</u>
- Demo: <u>https://naiquevin.github.io/rustick/</u>

Webassembly

<u>https://webassembly.org/</u>

Rust + Wasm

- <u>https://rustwasm.github.io/docs/book/</u>
- <u>https://rustwasm.github.io/wasm-bindgen/introduction.html</u>
- <u>https://developer.mozilla.org/en-US/docs/WebAssembly/Rust_to_Wasm</u>

Metronome & WebAudio API

- <u>https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API</u>
- https://grantjam.es/creating-a-simple-metronome-using-javascript-and-the-web-audio-api/

Thank you!