🚀 Supercharge your NodeJS with Rust

...

Dmitry Kudryavtsev
Senior Software Engineer @ Forter / ex-Autodesk
https://yieldcode.blog

What the…

JavaScript?!

JavaScript and NodeJS are great

But they are also slow

For some operations

Native Modules in C/C++ or Rust

But why Rust?

Why not C/C++?

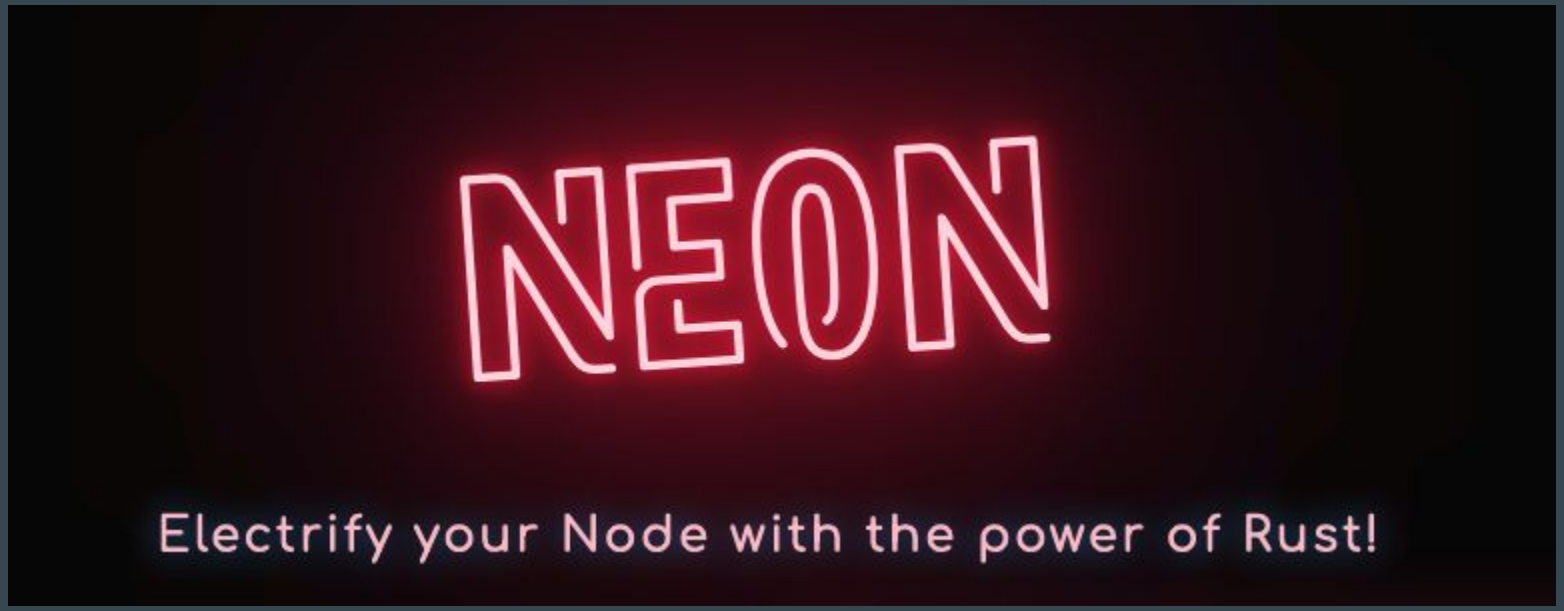
- They show their age
- They lack modern tooling (decent dependency manager, relatively poor stdlib)
- They are not memory safe (segfaults 😨)

Why Rust?

- Strongly typed & Compiled
- Rich stdlib: smart pointers, containers, iterators…
- Modern tooling: Cargo
- Memory safe (no segfaults 🤩)

Great and all… But… How?

Meet NEON. Node native modules with Rust



src/lib.rc

Require the needed code from Neon

The Fibonacci Logic

A “glue” layer between JS and Rust

The “export” of Rust function into JS world

```rust
use neon::context::{Context, ModuleContext, FunctionContext};
use neon::types::JsNumber;
use neon::result::JsResult;
use neon::result::NeonResult;

fn fibonacci(n: i32) -> i32 {
  return match n {
    n if n < 1 => 0,
    n if n <= 2 => 1,
    _ => fibonacci(n - 1) + fibonacci(n - 2)
  }
}

fn fibonacci_api(mut cx: FunctionContext) -> JsResult<JsNumber> {
  let handle = cx.argument::<JsNumber>(0).unwrap();
  let res = fibonacci(handle.value(&mut cx) as i32);
  Ok(cx.number(res))
}

#[neon::main]
fn main(mut cx: ModuleContext) -> NeonResult<()> {
  cx.export_function("fibonacci_rs", fibonacci_api)?;
  Ok(())
}
```

Building the Rust Module

```
cargo-cp-artifact -nc index.node -- cargo build
--message-format=json-render-diagnostics
```

A dynamic library file (.dll / .so equivalent from C/C++ under Windows / *nix)

How to call from NodeJS

```
1   const {fibonacci_rs} = require("./index.node");
2
3   const value = process.argv[2] || null;
4   const number = parseInt(value);
5
6   if(isNaN(number)) {
7     console.log("Provided value is not a number");
8     return;
9   }
10
11  const result = fibonacci_rs(number);
12  console.log(result);
```

I've heard something about so called WASM...

What is WASM / Web Assembly?

WebAssembly

- Portable Binary Format
  - And corresponding text format
- Executed by VM
- Supported in all major browsers (and NodeJS)
- Can be written in AssemblyScript
- A compilation target for other Languages (Rust among them)

src/lib.rs

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

#[wasm_bindgen]
fn fibonacci(n: i32) -> i32 {
    return match n {
        n if n < 1 => 0,
        n if n <= 2 => 1,
        _ => fibonacci(n - 1) + fibonacci(n - 2)
    };
}
```

What about Performance?

	30th Fibonacci	44th Fibonacci	45th Fibonacci	46th Fibonacci
JavaScript (NodeJS)	165.2ms	5.846s	9.358s	15.038s
Native Rust	161.5ms (+2.23%)	2.271s (+61.15%)	3.578s (+61.76%)	5.721s (+61.95%)
Rust WASM	163ms (+1.33%)	3.286s (+43.79%)	5.207s (+44.35%)	8.317s (+44.69%)

Run with Hyperfine tool, each run of Fibonacci was run with 3 warmups, taking the mean running time

Conclusion 1: Rust increases performance by ~60%. WASM by ~45%. (Compared to NodeJS)

Conclusion 2: Rust is ~45% faster than WASM

Note: Benchmarks like this are useless. Always run your own benchmarks

So… Native Modules or WASM?

Native modules or WASM?

Performance: Native Modules

- Native is always faster than VM (look at C/C++ vs Java)
- But WASM is still pretty fast!

Native modules or WASM?

Reusability: It’s Complicated

- Native library can be reused via FFI in other languages (Java, Swift)
- WASM is portable only across WASM VMs

Native modules or WASM?

Ergonomics: WASM

- WASM (bindgen) automatically converts basic types (i32, i64, f32, f64)
- NEON needs a “glue” layer to convert between Rust and JS

Native modules or WASM?

stdlib: Native Modules (or wait for WASI)

- WASM doesn’t have access to stdlib - so no filesystem, networking or anything OS related
- Unless you have WASI (WebAssembly System Interface) which is still in development

Native modules or WASM?

Portability: WASM

I don't know, it works on my machine - said every developer

- Native Modules are host machine dependant
- WASM is run by a VM

Native modules or WASM?

NodeJS vs Browser: It’s complicated

- Native Modules can’t be used in the Browser. JavaScript has no support for FFI
  - NodeJS uses N-API to build native modules with stable ABI
- WASM can be run if there is a WASM VM (so NodeJS + all browsers, except for IE 🤦‍♀️)

Native modules or WASM? - Conclusion

Performance	Rust Native Modules
Reusability	It's complicated
Ergonomics	WASM
stdlib	Rust Native Modules
Portability	WASM
NodeJS vs Browser	It's Complicated

Native modules are meant to **extend** NodeJS with performant code

WASM meant to **replace** non performant JS pieces of code

Thank you!

@kudmitry

@skwee357

@skwee357