Rust your own V8

Rust 🧡 JavaScript

JS? In MY Rust??

- Yeah, why not
- Powerful and easy scripting
- Much easier than WASM
- Lua is stinky

Examples where it's useful

- Game scripting/modding
- Advanced automations in professional programs
- Plugin system for large apps
- Basically anything that requires flexibility and isn't too performance-critical

Tangent: why not Lua though?

- Arrays start at 1
- enough said

Oh and also:

- Syntax is less familiar compared to C-style languages
- No type-oriented tooling
- Less built-in features (e.g. no async)

What prompted all of this anyway?

[rix]

- Nix is a functional programming language for package/system configuration
- It's interpreted
- Someone decided it's a smart idea to transpile Nix to JS and run it that way
- Because it's written in Rust, it's called Rix
- I decided to help contribute

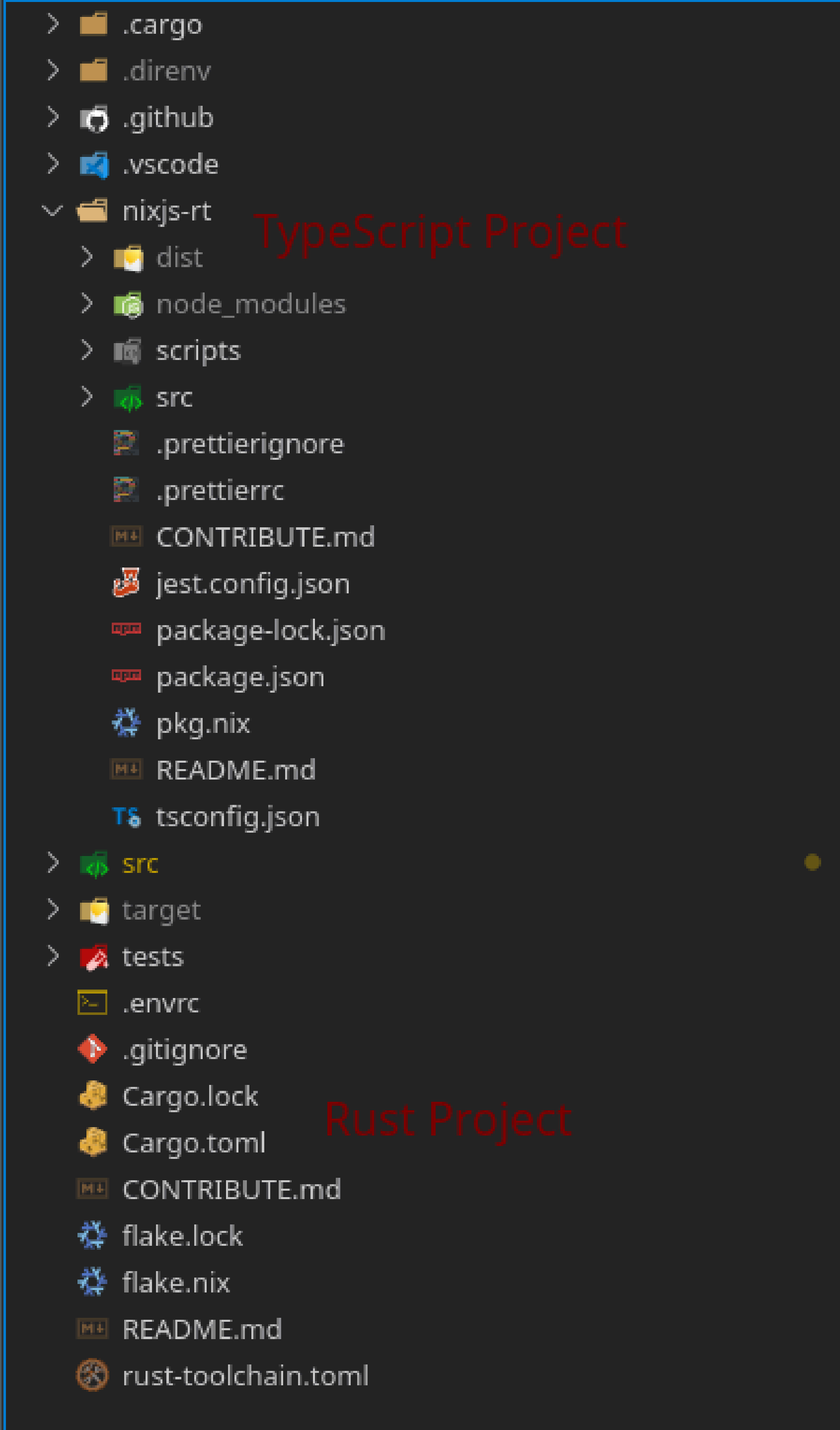
How I found Rix

- I've been using NixOS for a bit now (i use nix btw)
- Stumbled on rix, looked exactly like my kind of project
- Started with a small PR adding a couple builtin functions
- Added more PRs over time, until I got added as a contributor

Rix structure

Rix consists of 2 projects:

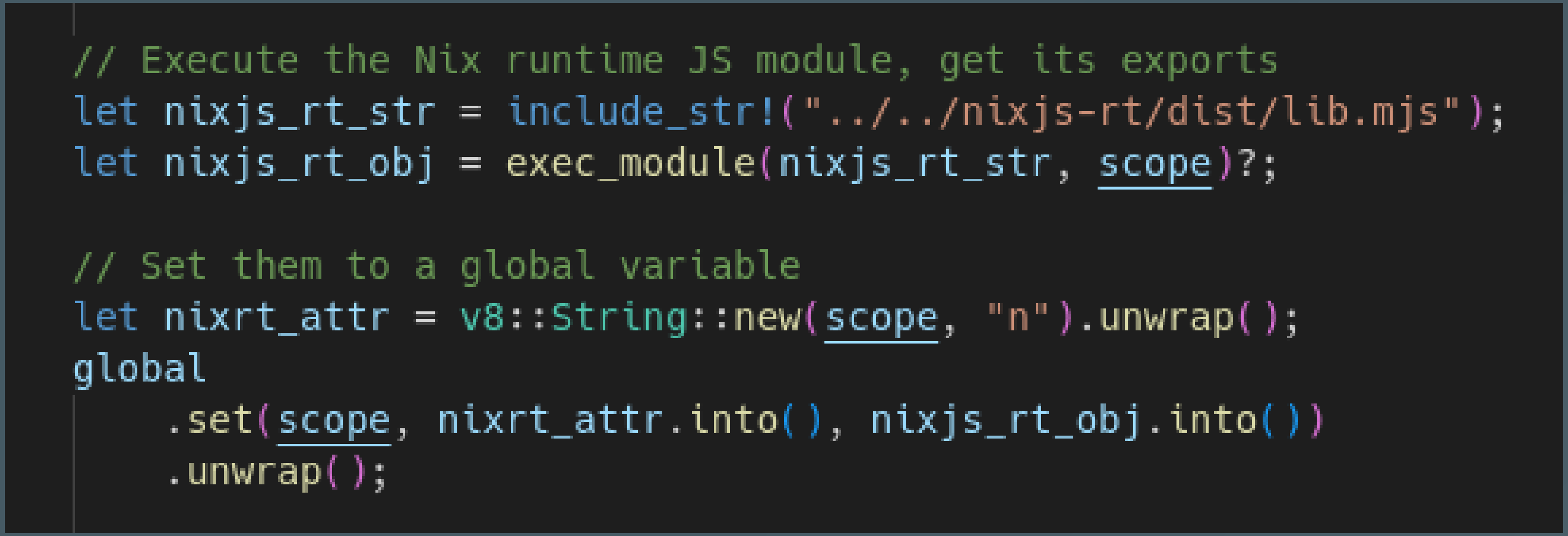
- The JS lib (compiled from TS)
- The Rust runtime, which uses V8
    - Imports JS using `include_str!`



```
> .cargo
> .direnv
> .github
> .vscode
v nixjs-rt
  > dist
  > node_modules
  > scripts
  > src
    .prettierignore
    .prettierrc
    CONTRIBUTE.md
    jest.config.json
    package-lock.json
    package.json
    pkg.nix
    README.md
    tsconfig.json
> src
> target
> tests
  .envrc
  .gitignore
  Cargo.lock
  Cargo.toml
  CONTRIBUTE.md
  flake.lock
  flake.nix
  README.md
  rust-toolchain.toml
```

Loading the JS lib

- Include the JS file as a string
- Execute it within V8
- Set the result to a global variable

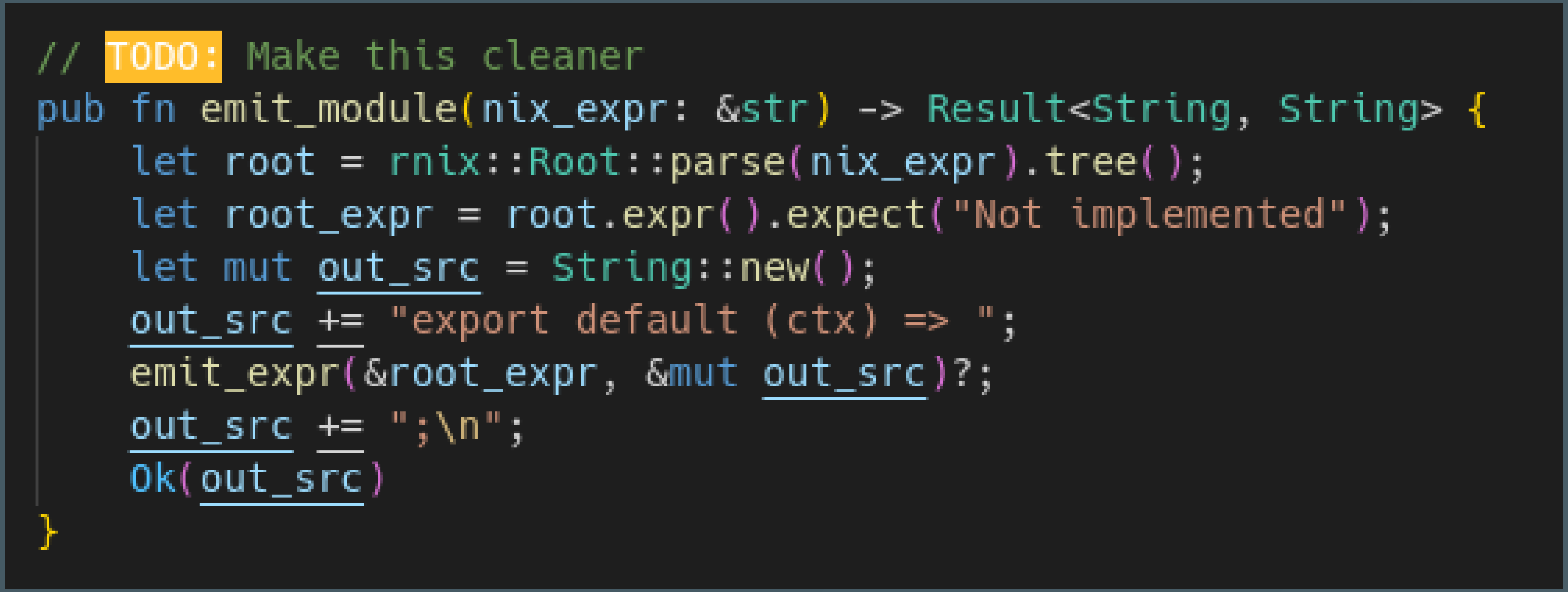


```
// Execute the Nix runtime JS module, get its exports
let nixjs_rt_str = include_str!("../../nixjs-rt/dist/lib.mjs");
let nixjs_rt_obj = exec_module(nixjs_rt_str, scope)?;

// Set them to a global variable
let nixrt_attr = v8::String::new(scope, "n").unwrap();
global
    .set(scope, nixrt_attr.into(), nixjs_rt_obj.into())
    .unwrap();
```

Building the code

- Transpile the Nix code to JS code
- Yes, this is not clean



```rust
// TODO: Make this cleaner
pub fn emit_module(nix_expr: &str) -> Result<String, String> {
    let root = rnix::Root::parse(nix_expr).tree();
    let root_expr = root.expr().expect("Not implemented");
    let mut out_src = String::new();
    out_src += "export default (ctx) => ";
    emit_expr(&root_expr, &mut out_src)?;
    out_src += ";\n";
    Ok(out_src)
}
```

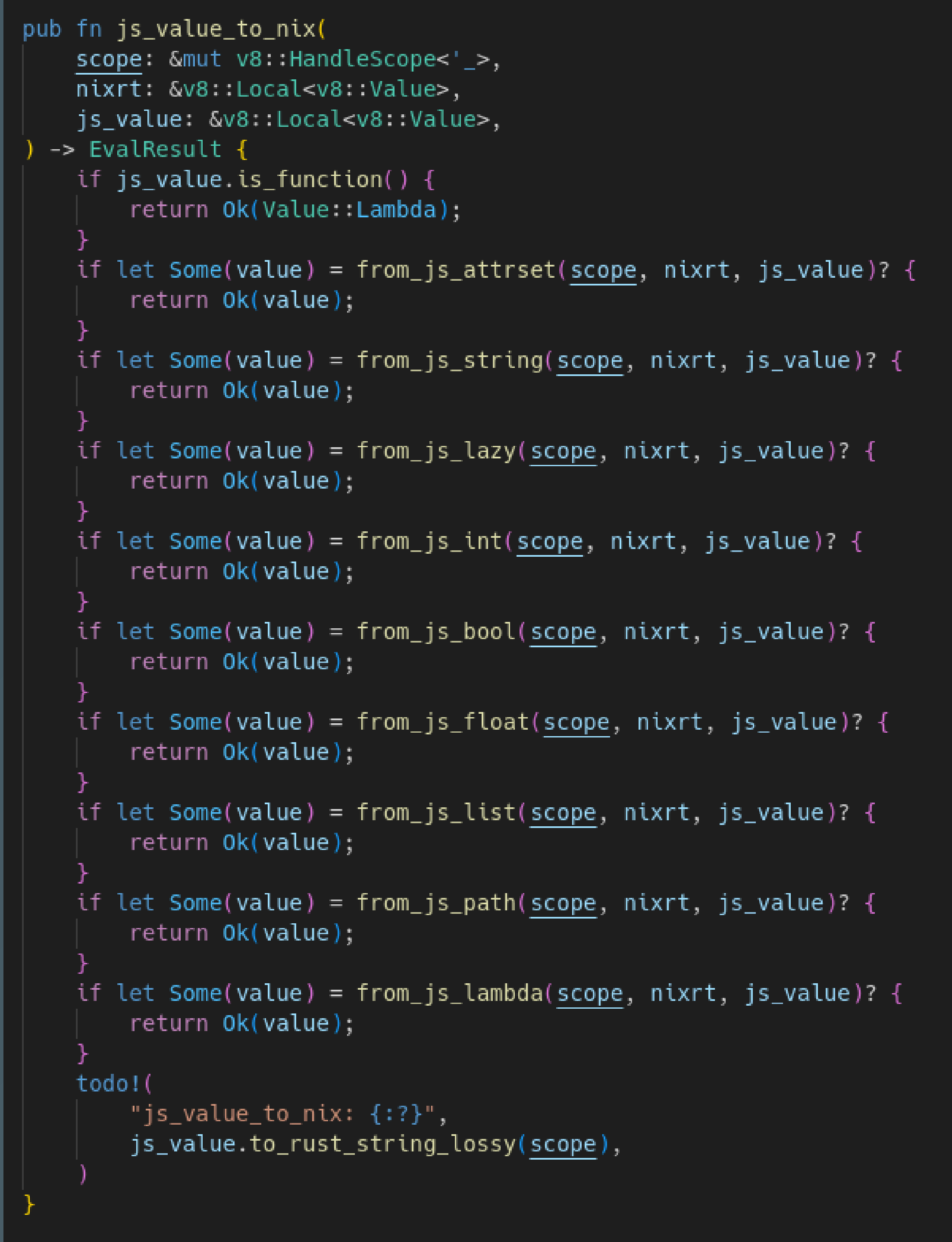
Executing the code

Too much code to show, but

- Take the transpiled code, execute it in V8
- Any "imports" performed in nix are also transpiled and converted to JS values

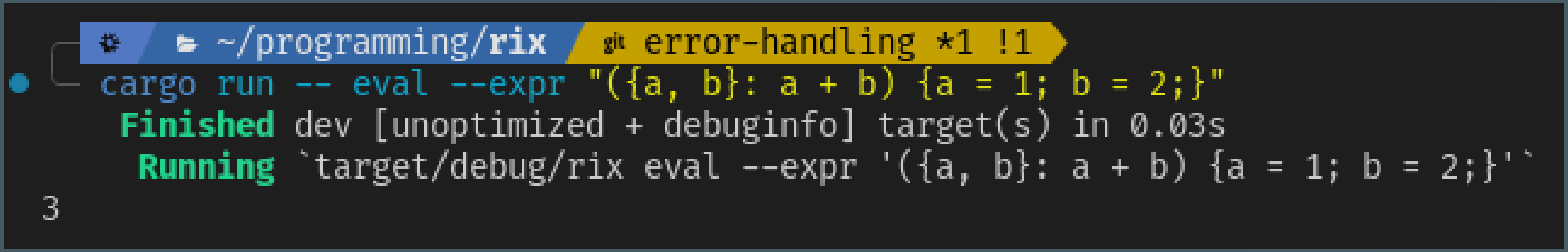
Parsing the results

- Given the results, convert them to Rust values
- Recursively iterate through all the JS classes, checking their instance type to know how to convert them

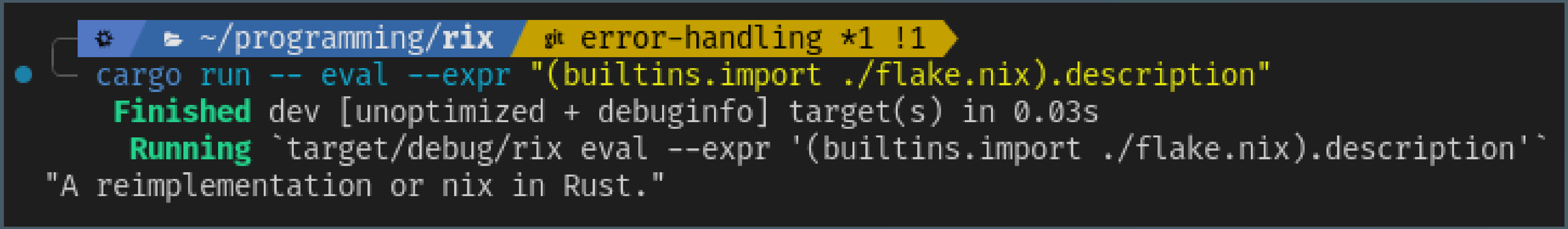


```rust
pub fn js_value_to_nix(
    scope: &mut v8::HandleScope<'_>,
    nixrt: &v8::Local<v8::Value>,
    js_value: &v8::Local<v8::Value>,
) -> EvalResult {
    if js_value.is_function() {
        return Ok(Value::Lambda);
    }
    if let Some(value) = from_js_attrset(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_string(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_lazy(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_int(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_bool(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_float(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_list(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_path(scope, nixrt, js_value)? {
        return Ok(value);
    }
    if let Some(value) = from_js_lambda(scope, nixrt, js_value)? {
        return Ok(value);
    }
    todo!(
        "js_value_to_nix: {:?}",
        js_value.to_rust_string_lossy(scope),
    )
}
```

Profit



```
 ~/programming/rix  error-handling *1 !1
cargo run -- eval --expr "({a, b}: a + b) {a = 1; b = 2;}"
    Finished dev [unoptimized + debuginfo] target(s) in 0.03s
     Running `target/debug/rix eval --expr '({a, b}: a + b) {a = 1; b = 2;}'`
3
```



```
 ~/programming/rix  error-handling *1 !1
cargo run -- eval --expr "(builtins.import ./flake.nix).description"
    Finished dev [unoptimized + debuginfo] target(s) in 0.03s
     Running `target/debug/rix eval --expr '(builtins.import ./flake.nix).description'`
"A reimplementation or nix in Rust."
```

How can you do this at home?

I made a basic repo that you can use as a starter:

v8-demo

Remind me to send this in Discord too

Thanks!