The Beauty of Rust

Cargo

Peter Zdankin

More than just programming

- Reproducible builds
- Test Suite
- Documentation
- Package Management
- Compilation
- Style guide
- Code improvement

Test Suite

- Unit/Integration tests are important
- Being able to effortlessly write tests is important
- In most languages you have an external test suite

- In rust: cargo test

```
$ cargo test

running 2 tests
test tests::test_bad_add ... FAILED
test tests::test_add ... ok

failures:

---- tests::test_bad_add stdout ----
        thread 'tests::test_bad_add' panicked at 'assertion failed: `(left == right)`
  left: `-1`,
 right: `3`', src/lib.rs:21:8
note: Run with `RUST_BACKTRACE=1` for a backtrace.


failures:
    tests::test_bad_add

test result: FAILED. 1 passed; 1 failed; 0 ignored; 0 measured; 0 filtered out
```

```rust
pub fn add(a: i32, b: i32) -> i32 {
    a + b
}

// This is a really bad adding function, its purpose is to fail in this
// example.
#[allow(dead_code)]
fn bad_add(a: i32, b: i32) -> i32 {
    a - b
}

#[cfg(test)]
mod tests {
    // Note this useful idiom: importing names from outer (for mod tests) scope.
    use super::*;

    #[test]
    fn test_add() {
        assert_eq!(add(1, 2), 3);
    }

    #[test]
    fn test_bad_add() {
        // This assert would fire and test will fail.
        // Please note, that private functions can be tested too!
        assert_eq!(bad_add(1, 2), 3);
    }
}
```

Documentation

- There are many schools of thought regarding documentation
- The best documentation is no documentation
- The worst documentation is no documentation

- Cargo doc —open

Package Management

- Use other peoples work
- Automatically download code, tests, documentation for packages
- Use crates.io to find packages to include in your project
- Enter dependencies in your Cargo.toml

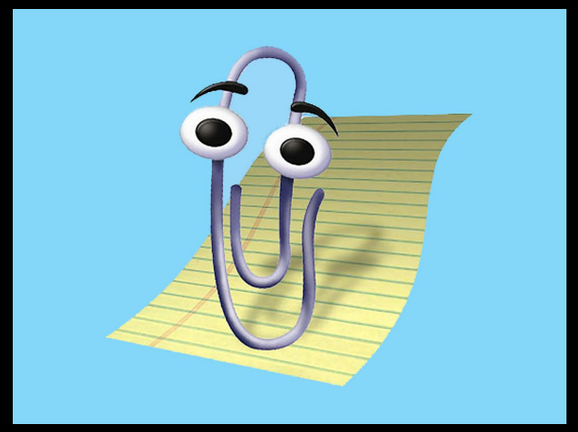
Compilation

- Everybody hates Makefiles
- Having a standard way to compile code to a binary is a game changer
- Most low-level languages (C/C++) are painful in this aspect
- Cargo build (—release)

Style guide

- Rust enforces a nice style guide
- All code has the same style guide, no project dependent manners
- Rustfmt

Code improvement



- Clippy helps you with writing better code
- Common pitfalls, problems etc
- Cargo clippy

That’s it from me!

Any Questions?