Rust Quick Reference

1. **Overview**

1.1 Characteristics

* Precompiled just like C.
* zero-cost abstractions: higher-level features that compile to lower-level code as fast as code written manually.
* Includes the official building system & packet manager **Cargo**, allowing the user to control and build dependencies.

1.2 Developer tools in Rust

* Cargo: the included dependency manager and build tool, makes adding, compiling, and managing dependencies painless and consistent across the Rust ecosystem.
* Rustfmt: ensures a consistent coding style across developers.
* The Rust Language Server: powers Integrated Development Environment (IDE) integration for code completion and inline error messages.

1.3 Fields on Rust

Command line tools, web services, DevOps tooling, embedded devices, audio and video analysis and transcoding, cryptocurrencies, bioinformatics, search engines, Internet of Things applications, machine learning, and even major parts of the Firefox web browser.

|  |  |
| --- | --- |
| Instruction | Description |
| Cargo new {name} | Generates the packet manager folder for Cargo to manage your rust project |
| Cargo build  --release | Compiles rust program.  Compiles with optimizations(superfast code however is slower compilation time) |
| ./target/debug/{CargoFile} | Creates executable of the cargo project on the target |
| Cargo run |  |
| Cargo check | Check correct compiling without producing an executable(speed-up the process) |

Variables

|  |  |
| --- | --- |
|  |  |
| let | Create a variable |
| mut | Assigns mutable(modifiable, non-static content) attribute. Eg: let mut guess = 5; |
| instance::method() | ´´method´´ is an associated function of ´´instance´´ type (static method).  Eg: let guess = String::new(); |
| instance::method.submethod() | Calls submethod on method handle. Eg: io::stdin().read\_line() |
| {} | Format specifiers(%) of rust. In Rust they are just a placeholder.  Eg:println!("x = {} and y = {}", x, y); |

Reference

|  |  |
| --- | --- |
|  |  |
| &mut | Indicate that the variable is a reference, which allows code access of one variable through memory location. |

Handling potential failures

|  |  |
| --- | --- |
|  |  |
| .expect(“failed”); | Handles errors when returned values give err (Result types return either “ok” or “err”). this crashes the program when an error occurs; Right way to recover from an error, is to write an error handling function. |

Crate:collection of Rust source code files(Basically a library).

Version: you can add dependencies with **version** included.Ensuring **reproducible builds**: Rust will not automatically upgrade your dependencies until you explicitly upgrade them, this maintains code compatibility. To ignore manual versions, use cargo update which will update dependencies to latest versions.