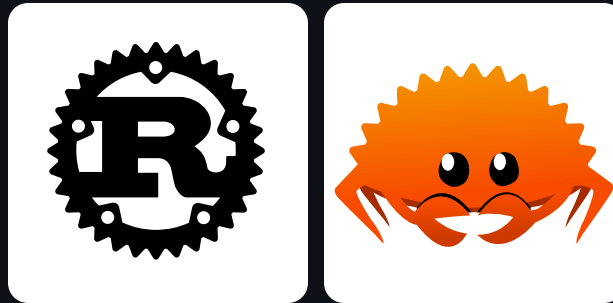


Introduction to the **Rust** programming Language



Following along [The Rust Book](#) from the official source

by: **Simon Lalonde**

For: **IFT-769** (Theoretical concepts CS)

Project overview - Going through "The Rust Programming Language"

The Rust Programming Language by Steve Klabnik and Carol Nichols



Book overview:

- Official guide to the Rust programming language
- Covers the basics (syntax, types, functions)
- Build tool and package manager (Cargo)
- Advanced and Rust-specific features:
 - Ownership, borrowing, lifetimes
 - Unique error handling
 - Concurrency



Theoretical concepts - Key topics covered

1. Common Programming Concepts (*variables, types, control flow*)
2. Understanding **Ownership** (*memory management*)
3. Structs, Enums and Pattern Matching
4. Containers/Collections
5. **Error Handling**
6. **Generics, Traits and Lifetimes**
7. Functional and OO features
8. **Smart pointers and Concurrency**
9. Patterns and matching + Advanced features

Klabnik, Steve, and Carol Nichols. The Rust Programming Language. 2nd ed., No Starch Press.



Practical project #1 - Write an I/O CLI program

Halfway project for a `grep` clone CLI app covers:

1. Code organization (crates, modules)
2. Use of containers and strings
3. Error handling
4. Using traits and lifetimes
5. Testing and documentation

Klabnik, Steve, and Carol Nichols. The Rust Programming Language. 2nd ed., No Starch Press.



Practical project #2 - Building a Multithreaded Web Server

Final Project from the book includes :

1. Learn TCP/IP networking and HTTP
2. Listen to TCP connections on a socket
3. Parse HTTP requests
4. Generate HTTP responses
5. Handle multiple requests concurrently with a thread pool

Klabnik, Steve, and Carol Nichols. The Rust Programming Language. 2nd ed., No Starch Press.

Rust Overview

- Systems programming language focused on safety and performance
- TODO

Currently known projects

TODO

Predicted use cases

TODO



Pros and Cons of Rust

PROS:

- **Memory safety:** No null pointers, dangling pointers, or buffer overflows
- **Error handling:** With the `Result` and `Option` types
- **Concurrency:** Safe and efficient with the ownership system
- **Performance:** Comparable to C/C++ with zero-cost abstractions
- **Ecosystem:** Growing with a strong community and package manager (**Cargo**)
- **Helpful compiler:** Provides detailed error messages and warnings

CONS:

- **Learning curve:** Ownership, borrowing, and lifetimes can be challenging
- **Tooling and prevalence:** Not as mature as other languages (C/C++, Python, etc.)
- **Syntax:** Can be verbose and complex compared to other languages



Installation and setup

Installation

1. Install **Rust** using `rustup` (Rust toolchain installer)

Package and library management

- **Crates** are Rust packages that can be shared and reused
- Managed with **Cargo**, the Rust package manager



Installation and Cargo/Crates example



DEMO



EXAMPLE TITLE

TODO



GREP PROJ? TODO

TODO

