Module 1: Introduction

Lesson 1.1: Introduction

Rust is a modern systems programming language pursuing the trifecta of safety, fearless concurrency, and blazingly fast speed, making it distinct from older systems languages like C++. Originating as a personal project in 2006, Rust was sponsored by Mozilla starting in 2009 and is now stewarded by the independent Rust Foundation created in 2021.

Key Takeaways

- Rust aims to achieve the high speed and concurrency of C/C++ while providing safety guarantees usually associated with high-level languages.
- Learning Rust requires active participation, including completing exercises and creating personal projects, especially due to its initially steep learning curve.
- The core of Firefox was rewritten in Rust, leading to the Firefox Quantum update in 2017, which resulted in significant speed improvements and fewer bugs.
- The Rust Foundation, funded by industry giants like AWS and Google, is an independent non-profit stewarding the language and ecosystem.

Key Concepts

- **Trifecta (Safety, Concurrency, Speed):** Rust pursues guaranteed safety at compile time, leading to easier concurrency, and achieves blazing speed through features like zero cost abstractions.

- **Systems Programming Language:** A language designed for low-level infrastructure tasks, but Rust provides modern safety features lacking in traditional systems languages.

- **Steep Learning Curve:** New users must learn the fundamentals of Rust thoroughly, as failing to understand them often prevents code from compiling successfully.

Lesson 1.2: Exercises Overview

This course provides a companion repository on GitHub, located under the username "CleanCut" with the repository name "UltimateRustCrashCourse". Students should clone this repository, where each project inside the exercise sub-directory is a standalone Rust project to be completed.

Key Takeaways

- The companion repository, located on GitHub, contains the set of practical exercises for the course.
- Each exercise should be opened in an IDE or editor, and instructions are found in the src/main.rs file.
- Cloning or downloading the repository is necessary to access the standalone Rust projects.

Key Concepts

- **Companion Repository:** The dedicated GitHub source for course materials, containing standalone Rust projects that facilitate learning by doing.
- **Standalone Rust Project:** Each exercise is a complete, independent Rust project located in its own sub-directory, ready for compilation and running.

Lesson 1.3: Installation

RustUp is the recommended tool for installing and managing Rust, allowing users to handle multiple versions and easily keep their installation up to date. Users should visit rustup.rs to follow the installation instructions and should use the current stable version of Rust, which RustUp installs by default.

Key Takeaways

- RustUp is the primary management tool, responsible for keeping Rust and important tools updated.
- If default installation options are accepted, RustUp automatically installs the current stable version.
- The rustup update command is used to update Rust to the latest stable version.

Key Concepts

- **RustUp:** The essential tool for managing Rust installations, offering version control and update functionality.
- **Stable Version:** The recommended version for the course; new releases occur at least every six weeks.
- **rustup update**: The command necessary to pull and install the newest stable version of Rust.

Lesson 1.4: Editor/IDE

The recommended tool for enabling Rust support in most editors and IDEs is Rust Analyzer, which provides features like GoToDefinition and IntelliSense via the Open Source Language Server protocol. JetBrains IDEs are the main exception, as they use the proprietary IntelliJ Rust plugin, which is a serious competitor to Rust Analyzer.

Key Takeaways

- Rust Analyzer is necessary for optimal development experience; users should consult its website for specific installation instructions.
- Conversions between types that might fail use methods like try_into.
- try_into returns a Result type, requiring error handling.

Key Concepts

- **Rust Analyzer** is necessary for optimal development experience; users should consult its website for specific installation instructions.
- VS Code users must use the **Rust Analyzer plugin** and avoid the older "Rust" plugin.
- A Tommel plugin is also recommended to check dependency versions within configuration files.
- For users without a preference, the recommendation is to download **VS Code** and install Rust Analyzer and a Tommel plugin.