The Rust Programming Language

Modality: On Demand

Duration: 5 Hours

About this course:

More and more often these days, we're hearing news reports of large-scale security threats to highly sensitive computer systems. Whether it's a major website getting hacked, personal information stolen from company databases or even cyber identity theft, there is always one common weak link; unsafe code. Enter Rust; the brainchild of Mozilla that promises fast, efficient, and memory safe systems programming. Sound intriguing? This course will teach you everything you need to know.

What is Rust?

Rust is a general purpose, low level programming language created by Mozilla. It was designed as a systems programming language with an emphasis on being safe, concurrent and practical. Its syntax is similar to C++, but it practices better memory safety while maintaining performance. Developers praise it for its speed and safety. It was recently awarded the title of 'most loved programming language' in the Stack Overflow Developer Survey.

Course Objective:

- Learn to program in Rust
- Understand Rust's memory management abilities
- Get to grips with a systems programming language
- Create highly concurrent, safe systems
- Increase your coding skill set

Systems Programming Made Safe:

This course is aimed at intermediate coders with some previous programming knowledge. If you're comfortable working with a code editor and want to add another tool to your programming skill set, then this course is perfect for you.

After starting with an extensive overview of Rust fundamentals, you'll hit the ground running and dive into more advanced features like vectors, generics, tuples, ownership borrowing and crates. Module 'challenges' throughout the course put your knowledge to the test – but if you need a push in the right direction, video walk throughs and code downloads are available.

By the end of this course you'll have a thorough understanding of Rust, of memory management in general and of systems programming specifically. You'll be well placed to begin or further your knowledge of similar languages like C++, and you can impress potential employers with your expertise in safe code.

Course Outline:

Introduction to the Course

- Course Introduction (2:44)

Environment

- Introduction (2:18)
- Introduction to Rust and Installation (7:32)
- Editor (6:46)
- Summary (1:06)

Language Fundamentals I

- Section Introduction (3:45)
- Data Types (8:59)
- More Data Types and Memory Size (11:25)
- Stack and Heap (9:12)
- Scope and Shadowing (9:00)
- Arithmetic and Conditional Operators (9:48)
- M2 Challenge (5:18)
- Section Summary (1:35)

Language Fundamentals II

- Section Introduction (1:29)
- Conditionals (10:32)
- Loops (10:11)
- Match (6:48)
- M3 Challenge (5:48)
- Section Summary (1:15)

Option & Vectors

- Section Introduction (2:34)
- Structs and Enumeration (6:21)
- Option (5:57)
- Option (advanced topics) (6:46)
- Arrays (9:12)
- Vectors (5:05)
- M4 Challenge (6:43)
- Section Summary (1:08)

Strings, Tuples, Functions

- Section Introduction (1:14)
- Slicing (7:09)
- String Manipulation (8:15)
- String Challenge (3:40)

- Tuples and Structs (6:33)
- Tuples and Structs Challenge Part1 (10:34)
- Tuples and Structs Challenge Part 2 (10:06)
- Functions and Closures (8:52)
- Section Summary (1:02)

Ownership, Borrowing, Crates

- Section Introduction (1:23)
- Ownership Part 1 (10:23)
- Ownership Part 2 (5:58)
- Borrowing Part 1 (10:15)
- Borrowing Part 2 (6:57)
- Creating A Crate From Scratch (10:48)
- Modifying Open Source Crate (5:56)
- Section Summary (0:50)
- Course Summary (1:52)