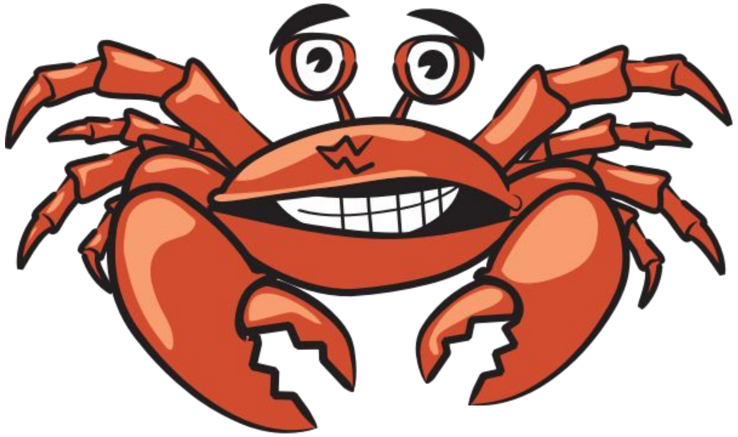
Introduction To Rust

Overview of Rust

Rust: A systems programming language focused on speed, memory safety, and parallelism.

- Created By: Graydon Hoare at Mozilla Research.
- First Released: 2010, stable release in 2015.
- Purpose: Designed to be a safe, concurrent, and practical language.
- Usage: Ideal for systemlevel programming, web development, and more.



The Evolution of Rust

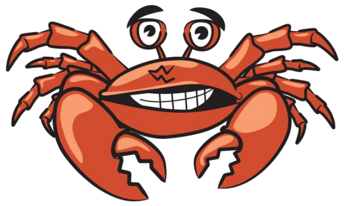
- 2006: Graydon Hoare begins development.
- 2009: Mozilla begins sponsoring Rust's development.
- 2010: Rust publicly announced.
- 2015: Rust 1.0 released.
- 2016: Continual improvements, growing community and ecosystem.

Unique Features

- Memory Safety: Rust makes sure your program doesn’t crash or misbehave by preventing common mistakes related to memory, like accessing invalid memory locations or overwriting important data.
- Ownership and Borrowing: Think of ownership like borrowing a book from a library. Only one person can borrow the book at a time, and you must return it before someone else can use it. This helps Rust keep track of who is using what part of the computer’s memory, preventing mistakes when multiple parts of a program try to use the same memory at once.
- Concurrency: Concurrency means doing multiple things at the same time. Rust makes it easier to write programs that do this safely, avoiding common problems like data getting mixed up when different parts of the program are running at the same time.
- Pattern Matching: Pattern matching is a feature that helps you write cleaner and more understandable code by allowing you to handle different possible values in a clear and organized way.
- Cargo: Rust's package manager and build system, Cargo, simplifies project management, dependency management, and compilation.

Advantages Over Other Languages

- Safety: Unlike C and C++, Rust guarantees memory safety without a garbage collector, reducing the risk of crashes and security vulnerabilities.
- Performance: Rust's performance is comparable to C and C++, making it suitable for high performance applications.
- Concurrency: Rust's unique approach to concurrency helps avoid data races, which are common in languages like C++ and Java.
- Community and Ecosystem: Rust has a strong and growing community with a rich ecosystem of libraries and tools, making it easier to find support and resources.
- Tooling: The Rust compiler and tools like Cargo provide excellent support for development, testing, and deployment, enhancing productivity.



Thank you for your time 😊