Introduction

L: Rust began development in 2006 by researchers at Mozilla research, and a stable version was first released in 2015.
It is multi-paradigm, as it's both functional and imperative.
Rust, as a language, prioritizes speed and security, not using runtime or any sort of garbage collection, and doesn't allow null or dangling pointers. There is also an emphasis on ownership and scope of variables.

Why Use Rust? (Features and Distinctions from other Languages)

J: As StackOverflow's "most loved language" for four years in a row now, Rust is clearly well-beloved by those who use it. Why is that? Below are some reasons as to why someone might want to develop in Rust.

J: There is a lot of debate about the preference between statically and dynamically typed languages. While dynamically typed languages are less verbose and easier to understand, statically typed languages allow for faster debugging and runtime efficiency. Rust tries to mitigate the tediousness of other statically typed languages while maintaining their benefits by requiring top-level items (think function arguments and constants) to have explicit types, while allowing type inference in function bodies. The compiler can then infer the types of variables in function bodies from the typing of the function signature.

L: To increase the speed of runtime, expedited by garbage-collection being performed (i.e. unnecessary memory gets deleted) at compile time, as opposed to runtime. This also saves on memory usage, and saves time on memory access.

J: To further increase speed, Rust allows developers to have control over low-level details, similar to other systems programming languages like C and C++. For example, developers can choose whether to store data on the stack or the heap. However, Rust has features that are not available in other systems programming languages. For example, Rust tries to have as many abstractions as possible which still are just as efficient as the non-abstracted code. Rust calls these *zero cost abstractions.*

L: Safe Rust is designed to help the user avoid undefined behavior, and essentially cleans up as the user goes along (again, at compile time), but users are allowed to write in unsafe Rust should they need more freedom in their code.

L: Rust was designed to be user-friendly, and fast to both write and run, making it invaluable for use in a variety of contexts, from systems programming to app design.

Categories: compiled, functional

Sources:

- https://www.rust-lang.org/
- https://doc.rust-lang.org/book/
- https://stackoverflow.blog/2020/01/20/what-is-rust-and-why-is-it-so-popular/#:~:text=One%20of%20the%20biggest%20benefits,and%20can%20be%20cleaned%20up.

Cute error messages