Aubrey Fernando
Abby Tan
Chase Mitchell
John Park

Rust

WHY WE CHOSE RUST

We chose rust because it was the “most loved programming language” from a Stack Overflow developer survey every year since 2016. We thought it would be interesting to research a language that is not only syntactically similar to C++ but also memory efficient while maintaining high performance. Rust is able to maintain such a high performance because it has no runtime or garbage collector.

RUST HISTORY

Rust grew out of a personal project began by a Mozilla employee, Graydon Hoare. Eventually, in 2009, Mozilla sponsored the project and the first release of the Rust compiler was in 2012. A few versions and years later in 2014, Rust was referred to as a competitor to C++ and other upcoming languages such as D and Go. Since falling into third place in 2015, Rust took first place title as the most loved programming language from 2016.

SOFTWARE PROGRAMMED IN RUST

Rust is currently being used in a variety of systems and applications including embedded systems, IoTs, operating systems. Because of Rust’s strict compiler that ensures safe code, it is used mostly in many embedded electronics. Rust is a good option for embedded system devices because it can ensure that resources won’t be used by the unintended part of the application. Notably, Rust makes it impossible to accidentally share state between threads. This makes Rust a better option that C for embedded devices or systems that use threads because race conditions and deadlocks can be avoided. There are currently many web browser components that are written in Rust for Firefox. Servo is a Mozilla web browser engine that was purposely developed to take advantage of Rust’s memory safe capabilities to achieve greater security and performance than contemporary web browsers. WebAssembly is another example of a Rust implemented web component for the deployment of client and server applications. WebAssembly has the ability to run code at near native speeds and it is compiled by Rust, rather than C++. Rust is the best option for WebAssembly because Rust has such a minimal runtime.

WHY SHOULD YOU USE RUST / WHAT TASKS RUST IS SUITABLE FOR

There are many reasons why one would choose to use Rust. Rust is intended to be a language for highly safe systems while maintaining boundaries that preserve large-system integrity. It has an emphasis on safety, control of memory layout and concurrency. Rust is suitable for tasks that cannot afford errors or mistakes. The rust compiler is stricter and makes sure that it's using memory safely. Using Rust would reduce the amount of vulnerabilities in a program that could be exploited. In comparison to C++, 70% of its vulnerabilities and exposures (CVEs) are memory-related. As mentioned previous, Rust's concurrency capabilities make Rust not only a great option to use in embedded devices and IoT systems, but Rust also has excellent networking abilities. Rust's powerful compiler prevents a whole class of bugs and ensures you know when and where each state is shared and mutated. In addition, it has a low footprint that takes control of resource usage to keep memory and CPU footprint to a minimum.

INTERESTING FEATURES

Rust has many features that set it apart from other programming languages that stood out to us. The rust compiler is very strict and checks each variable and memory address you reference. Though it may seem like a hindrance in writing effective and expressive code, it makes programming easier by avoiding potentially writing unsafe code. Like Haskell, Rust has a way to somewhat prove the code. Though Rust cannot formally prove the code like Haskell, it's an impressive feature that Rust, as an imperative language is currently on it's the way to becoming formally provable.

Because of Rust's level of control, ability to catch memory and concurrency errors, Rust has positioned itself to be not only one of the most loved programming languages but also the most general-purpose programming languages that can be used to write secure and efficient programs.