Rust Fact Sheet
CSCI 315, Programming Languages

Gage Smith

Spring 2019

Overview

Rust is a programming language written by Graydon Hoare originally in 2006 while he was working for Mozilla Research. After its first stable release in 2015 it quickly became regarded as an open-source systems programming language. Rust focuses on speed, memory safety and parallelism. [3]

Version History

Current Version: 5.22.4, released July 15, 2017

Rust: Major Revision History[3]

Year	**Version**	**Note**
2006	Pre-Release	Mozilla employee Graydon Hoare began Rust as a personal project.
2009-2010	Pre-Release	Mozilla sponsored the project and announced it.
2011	Pre-Release	First successful compilation with self-hosting compiler written in Rust.
2015	1.0	First stable release of Rust v1.0.
2015	1.2	Addition of classes.
2015	1.3	Destructors and polymorphism through interfaces.
2015	1.4	Traits added to have a means of inheritance.
2017	1.15	Rust adopted the ”open roadmap process”.
2017	1.21	Firefox Quantum begins utilizing a pure Rust CSS engine.
2017-2018	1.22-1.31.1	Stack Overflow annual survey most loved programming language.

Basics of Rust Programming[1]

Declaring a Main()

```
fn main()
{
    //Any amount of Rust code including other function declarations
}
```

Intializing Variables

```
    // Declare a variable binding
    let a_binding;

    {
        //Simple form
        let x = 2;

        // Initialize the binding
        a_binding = x * x;
    }
```

Printing out Variables

```
    // In general, the `{}` will be automatically replaced with any
    // arguments. These will be stringified.
    println!("{} days", 31);

    // Without a suffix, 31 becomes an i32. You can change what type 31 is,
    // with a suffix.

    // There are various optional patterns this works with. Positional
    // arguments can be used.
    println!("{0}, this is {1}. {1}, this is {0}", "Alice", "Bob");

    // As can named arguments.
    println!("{subject} {verb} {object}",
             object="the lazy dog",
             subject="the quick brown fox",
             verb="jumps over");

    // Special formatting can be specified after a `:`.
    println!("{} of {:b} people know binary, the other half doesn't", 1, 2);
```

Code Examples[1]

Simple "Hello World"

```
// This is the main function
fn main()
{
    // The statements here will be executed when the compiled binary is called

    // Print text to the console
    println!("Hello World!");
}
```

Type Annotation and Primitives

```
fn main()
{
    // Variables can be type annotated.
    let logical: bool = true;

    let a_float: f64 = 1.0;  // Regular annotation
    let an_integer   = 5i32; // Suffix annotation

    // Or a default will be used.
    let default_float   = 3.0; // `f64`
    let default_integer = 7;   // `i32`

    // A type can also be inferred from context
    let mut inferred_type = 12; // Type i64 is inferred from another line
    inferred_type = 4294967296i64;

    // A mutable variable's value can be changed.
    let mut mutable = 12; // Mutable `i32`
    mutable = 21;

    // Error! The type of a variable can't be changed.
    mutable = true;

    // Variables can be overwritten with shadowing.
    let mutable = true;
}
```

Working With Block Expressions

```rust
fn main() {
    let x = 5u32;

    let y = {
        let x_squared = x * x;
        let x_cube = x_squared * x;

        // This expression will be assigned to `y`
        x_cube + x_squared + x
    };

    let z = {
        // The semicolon suppresses this expression and `()` is assigned to `z`
        2 * x;
    };

    println!("x is {:?}", x);
    println!("y is {:?}", y);
    println!("z is {:?}", z);
}
```

General If-Else Structure

```rust
fn main() {
    let n = 5;

    if n < 0 {
        print!("{} is negative", n);
    } else if n > 0 {
        print!("{} is positive", n);
    } else {
        print!("{} is zero", n);
    }
}
```

Useful Applications[2]

- Systems programming
- Command Line Interfaces (CLI)
- Web Assembly
- Utilizing type system for Networking interfaces
- Game Emulation/Game Development
- Package Management Software

Cargo Build System Tutorial/Overview[1]

Rust has a build system and package manager called ”Cargo”. Cargo allows for simple and easy downloading/building libraries for your Rust code to run upon.

```
#Check your version number
$ cargo --version

#Move your source files in to a "src" directory
$ mkdir src
$ mv main.rs src/main.rs
$ rm main

#Make a Cargo configuration file .toml
#Place this file in your project directory that contains the "src" directory
#Must have a capitalized 'C'

$touch Cargo.toml

#Add to Cargo.toml
[package]

name = "hello_world"
version = "0.0.1"
authors = [ "Your name <you@example.com>" ]

#Simple build and run commands with Cargo
$ cargo build
$ cargo run
```

Interesting Factoids[3]

1. Rust was originally named after a fungus called rust.

2. Rust was nominated ”most loved programming language” in Stack Overflow Developer Survey three years straight from 2016 to 2018.

3. There is a list of games* completely composed of Rust.

Quick Reference[3]

URL: `www.rust-lang.org`

Extension: `.rs .rlib`

Operating System: cross-platform

Written in: Rust/OCaml

Paradigm: multi-paradigm
(imperative, functional, concurrent)

Typing: Static, Strong, Linear

Appeared: 2010

Age in 2018: 8

Created by: Graydon Hoare

Influenced by: C#, C++, Haskell, OCaml, Scheme, Swift

Canonical Text : *Programming Rust: Fast, Safe Systems Development* (2017)

"You can't just reimplement everything in Rust"
that's where you're wrong kiddo

References

[1] Rust Team. Rust by example, 2019. https://doc.rust-lang.org/stable/rust-by-example/index.html.

[2] Rust Team. Rust language organization, 2019. https://rust-lang.org/.

[3] Wikipedia contributors. Rust (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Rust_(programming_language)&oldid=883708406, 2019. [Online; accessed 22-February-2019].