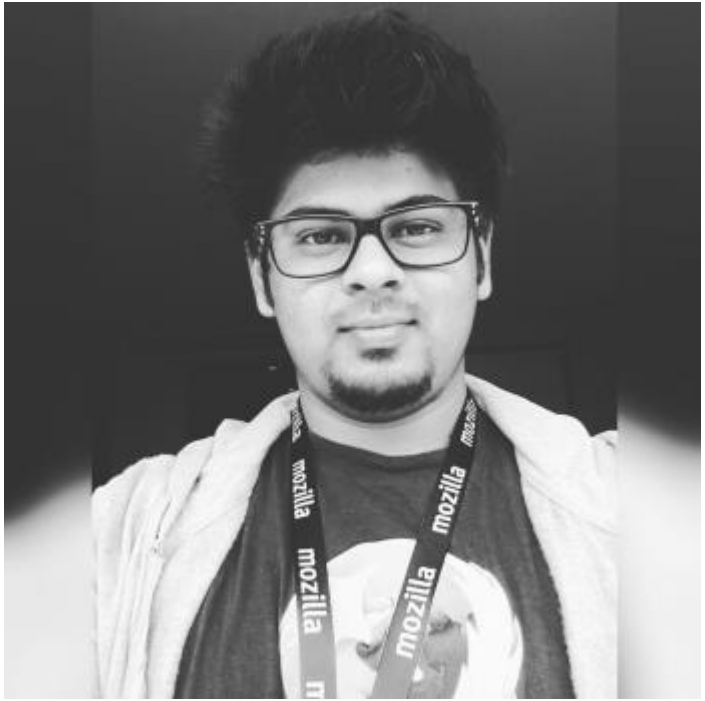
Rust: Introduction

@jayeshkattar

About Me

- Jayesh Katta Ramalingaiah
- Android Developer
- Work @tcs
- @mozilla Representative
- @mozilla Reps Mentor
- @mozilla Tech-Speaker



Agenda (for Introduction)

1. Basic Terminologies
2. What is Rust? Why Rust?
3. Intro to Rust
4. Install Rust
5. Clippy - the popular Rust static analysis tool
6. Cargo - Rust’s awesome package manager
7. Play with Rust: A short demo with kits.
8. How to get in touch with the Rust community?

Basic Terminologies

- Low and High Level Language
- System programming
- Concurrency and Parallelism
- Compile time and Run time
- Type System
- Garbage collector
- Mutability
- Scope

What is Rust?

- Rust is a new systems programming language designed for safety, concurrency, and speed.
- It was originally conceived by Graydon Hoare and is now developed by a team in Mozilla Research **and** the community.
- Functional, imperative, object-oriented, whenever it makes sense.
- Low-level. Targets the same problem-space as C and C++
- Safe. Lovely, lovely types and pointer lifetimes guard against a lot of errors.

Why do we need a new system programming language?

- State or art programming language
- Solves a lot of common system programming bugs
- Cargo : Rust Package manager
- Improving your toolkit

Rust

- System programming language
- Has great control like C/C++
- Safety and expressive like python

Why should I use Rust?

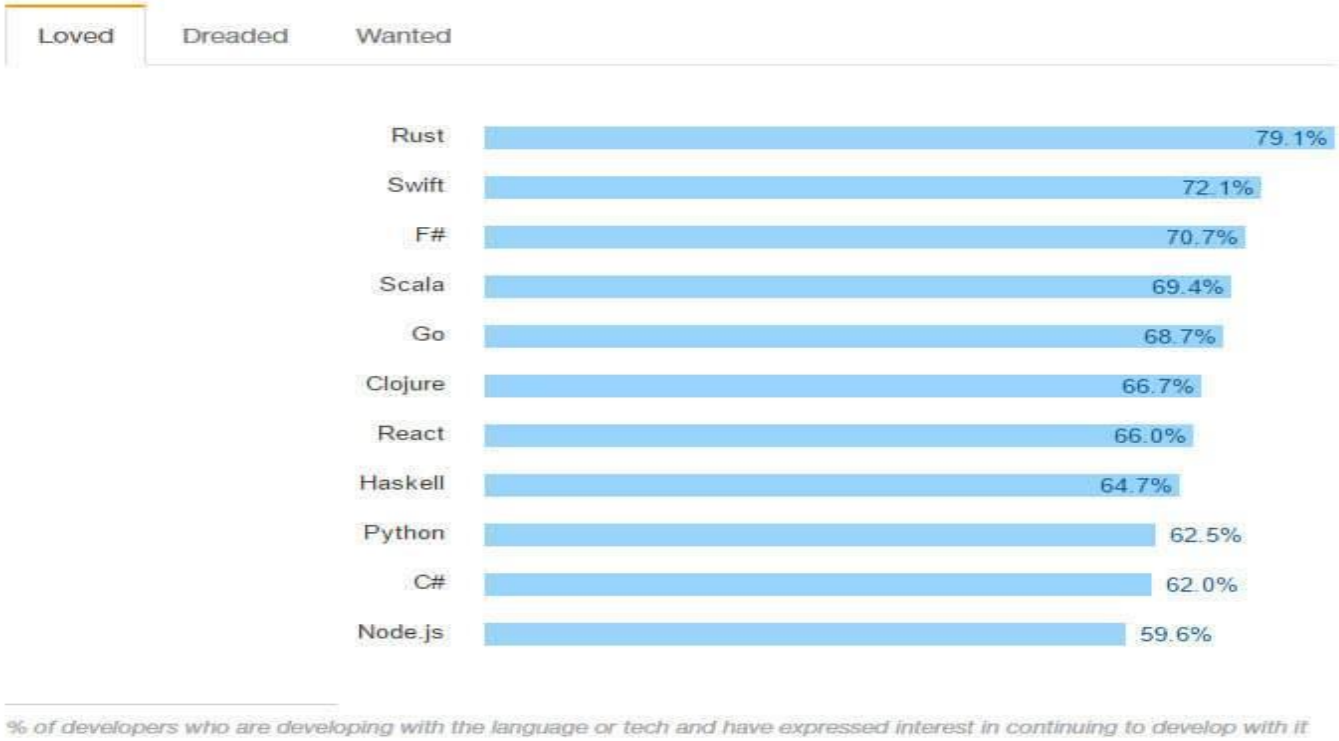
Own Definition :

Rust is a good choice when you’d choose C++. You can also say, “Rust is a systems programming language that pursuing the trifecta: safe, concurrent, and fast.” I would say, **Rust is an ownership-oriented programming language**.

The reason why Rust.

- Rust is new enough that you can write useful stuff that would have already existed in other languages
- It gives a relatively familiar tool to the modern C++ developers, but in the much more consistent and reliable ways.
- It is low-level enough that you take account of most resources.
- It's more like C++ and Go, less like Node and Ruby
- cargo is awesome. Managing crates just works as intended, which makes a whole lot of troubles you may have in other languages just vanish with a satisfying *poof*.

According to recent The Stack Overflow survey Rust is the most beloved among developers of all programming languages and frameworks.



Credits : https://insights.stackoverflow.com

Where is RUST used ?

- game engine (Piston library)
- game development
- browser engine (Servo)
- Build web apps
- html5ever (High-performance browser-grade HTML5 parser)
- tiny-http (http & https protocols)

Game Engine/ Game Development

- arewegameyet.com

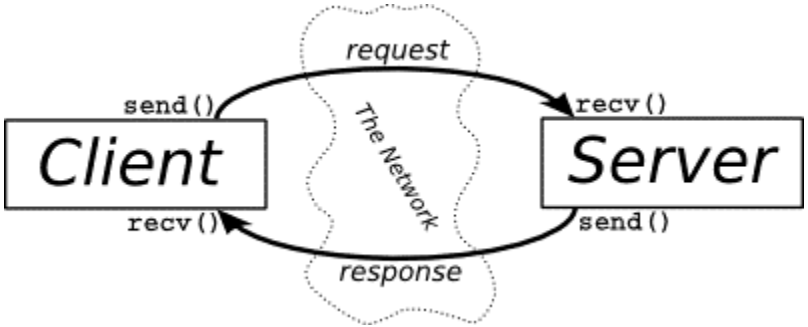
Gaming libraries in RUST which are used for game development.

Browser Engine

- Mozilla is building a new browser engine called SERVO.
- Servo is entirely written in RUST

Web Apps

- Web Developers even started using Rust for quicker responses.



Where can I get RUST?

- Prebuilt binaries are available at http://www.rust-lang.org/
- Source code is available from GitHub https://github.com/mozilla/rust
- Kits and resources are available from Rust India GitHub https://github.com/RustIndia/Rust

Install Rust with rustup

Ubuntu / MacOS

- curl -sSf https://static.rust-lang.org/rustup.sh | sh

Windows

- Go to https://win.rustup.rs/
  - This will download rustup-init.exe
- Double click and start the installation

Version

rustc --version

cargo --version

```
C:\Users\PingMe\hello_world>rustc --version
rustc 1.17.0 (56124baa9 2017-04-24)

C:\Users\PingMe\hello_world>cargo --version
cargo 0.18.0 (fe7b0cdcf 2017-04-24)
```

Cargo, Rust’s Package Manager

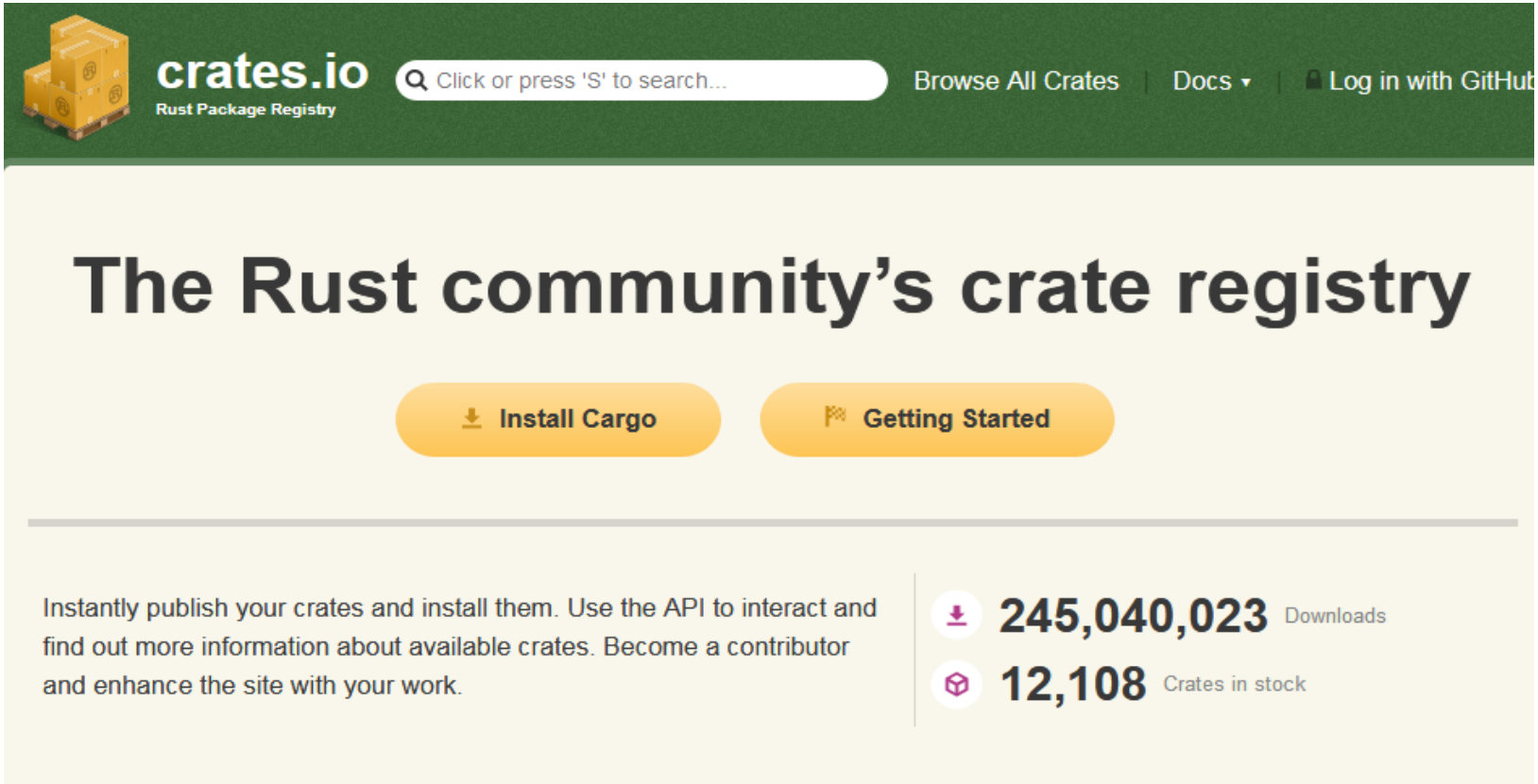
Cargo is a tool that allows Rust projects to declare their various dependencies and ensure that you’ll always get a repeatable build.

To accomplish this goal, Cargo does four things:

- Introduces two metadata files with various bits of project information.
- Fetches and builds your project’s dependencies.
- Invokes rustc or another build tool with the correct parameters to build your project.
- Introduces conventions to make working with Rust projects easier.

Creating a new project

https://crates.io/



Type System

The traditional Hello World

```
fn main() {
   let greet = “world”;
    println!("Hello {}!”, greet);
}
```

How to run ???

To compile and run a single file

to Compile: **rustc** *<filename>***.rs**

to run: **<***filename>*

Using Cargo package Manager

How to create a cargo package ??

Cargo new *<proj_name>* **--bin**

How to run a cargo package?

To see the tree structure of the package.

tree .

To compile and build a cargo package.

Cargo build

To run a cargo package.

Cargo run

Primitive Types

bool

```
let bool_val: bool = true;

println!("Bool value is {}", bool_val);
```

char

```
let x_char: char = 'a';

// Printing the character

println!("x char is {}", x_char);
```

Tuples

```
// Declaring a tuple

let rand_tuple = ("Mozilla Science Lab", 2016);

// tuple operations

println!(" Name : {}", rand_tuple.0);

println!(" Lucky no : {}", rand_tuple.1);
```

Arrays

```
let rand_array = [1,2,3]; // Defining an array

println!("random array {:?}",rand_array );

println!("random array 1st element {}",rand_array[0] ); // indexing starts with 0

println!("random array length {}",rand_array.len() );

println!("random array {:?}",&rand_array[1..3] );
```

String

```
let rand_string = "I love Mozilla Science <3"; // declaring a random string

println!("length of the string is {}",rand_string.len() ); // printing the length of the string

let (first,second) = rand_string.split_at(7);

println!("First {}",first );

println!("second {}",second );

let count = rand_string.chars().count();
```

Best things about Rust

- Strong type system
  - Reduces a lot of common bugs
- Borrowing and Ownership
  - Memory safety
  - Freedom from data races

Ownership

In Rust, every value has an “owning scope,” and passing or returning a value means transferring ownership (“moving” it) to a new scope.

<u>Example:</u>

```
fn foo{

    let v = vec![1,2,3];

        let x = v;

    println!(“{:?}”,v);

}
```

Ownership - Ex 2

```rust
fn make_vec() -> Vec<i32> {
    let mut vec = Vec::new();
    vec.push(0);
    vec.push(1);
    vec // transfer ownership to the caller
}

fn print_vec(vec: Vec<i32>) {
    // the `vec` parameter is part of this scope, so it's owned by `print_vec`

    for i in vec.iter() {
        println!("{}", i)
    }

    // now, `vec` is deallocated
}

fn use_vec() {
    let vec = make_vec(); // take ownership of the vector
    print_vec(vec);       // pass ownership to `print_vec`
}
```

```
fn print(v : Vec<u32>) {
    println!(“{:?}”, v);
}
fn make_vec() {
    let v = vec![1,2,3];
    print(v);
    print(v);
}
```

Borrowing

If you have access to a value in Rust, you can lend out that access to the functions you call

```rust
fn print_vec(vec: &Vec<i32>) {
    // the `vec` parameter is borrowed for this scope

    for i in vec.iter() {
        println!("{}", i)
    }

    // now, the borrow ends
}

fn use_vec() {
    let vec = make_vec();  // take ownership of the vector
    print_vec(&vec);       // lend access to `print_vec`
    for i in vec.iter() {  // continue using `vec`
        println!("{}", i * 2)
    }
    // vec is destroyed here
}
```

Rust Playground

- A web interface for running Rust code.
- The interface can also be accessed in most Rust-related channels on irc.mozilla.org.
- To use Playbot in a public channel, address your message to it.
  - `<you> playbot: println!("Hello, World");`
    `-playbot:#rust-offtopic- Hello, World`
    `-playbot:#rust-offtopic- ()`
    `<you> playbot: 1+2+3`
    `-playbot:#rust-offtopic- 6`
- You can also private message Playbot your code to have it evaluated. In a private message, don't preface the code with playbot's nickname:
  - `/msg playbot println!("Hello, World");`

Let’s play : https://play.rust-lang.org/

https://play.rust-lang.org

Run ASM LLVM IR MIR | Tools: Format Clippy | Share | Mode: Debug Release | Channel: Stable Beta Nightly | Config ?

```
fn main() {
    println!("Hello, world!");
}
```

Execution

Standard Error

```
Compiling playground v0.0.1 (file:///playground)
 Finished dev [unoptimized + debuginfo] target(s) in 0.45 secs
  Running `target/debug/playground`
```

Standard Output

```
Hello, world!
```

Let’s Meet Friends of Rust!!

Organizations running Rust in production
(https://www.rust-lang.org/en-US/friends.html)

Getting started with Rust community

- Follow all the latest news at Reddit Channel
  - https://www.reddit.com/r/rust/
- Have doubts, post in
  - https://users.rust-lang.org
  - #rust IRC channel
- Want to publish a crate,
  - https://crates.io
- Follow @rustlang in twitter,
  - https://twitter.com/rustlang
- Subscribe to https://this-week-in-rust.org/ newsletter

What Next ???

- https://github.com/jayeshkattar/Moz-Activate
- https://github.com/rustindia/Rust-for-undergrads
- https://github.com/rustindia
- https://github.com/MozillaIndia/RustIndia
- https://github.com/rusthacks/RustHack

Adopt Rust today !!

References

- Segfault: http://stackoverflow.com/questions/2346806/what-is-a-segmentation-fault
- BufferOverFlow: http://stackoverflow.com/questions/574159/what-is-a-buffer-overflow-and-how-do-i-cause-one
- Rust Website: https://www.rust-lang.org/en-US/
- Community Forum: https://users.rust-lang.org/
- Rust Book: https://doc.rust-lang.org/book/

Thank You

- Tweet at #RustLang
- Join RustIndia Telegram group

And Don’T forget to **TAG** me **@jayeshkattar**