**Why Rust?**

[[@imthemrluiz](https://devopsvision.medium.com/?source=post_page-----d8c9afa92e7f--------------------------------)](https://devopsvision.medium.com/?source=post_page-----d8c9afa92e7f--------------------------------)

[@imthemrluiz](https://devopsvision.medium.com/?source=post_page-----d8c9afa92e7f--------------------------------)

·

Follow

10 min read

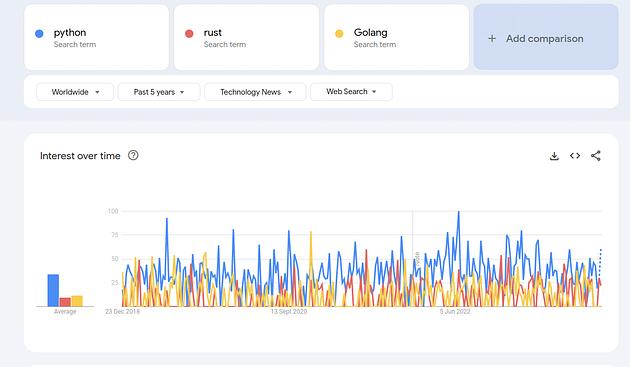
·

2 days ago

27

3

Performance, Reliability, and Productivity. Among many programming languages, why rust?

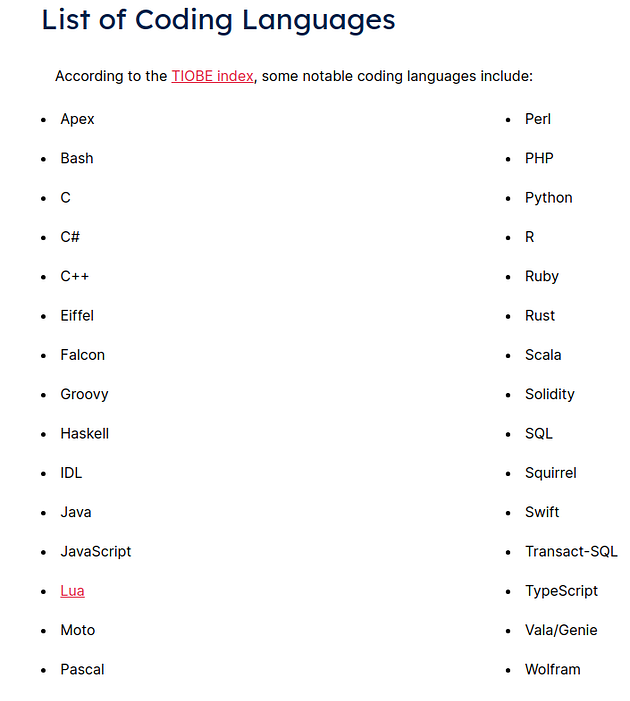


Google Trends — Python, Rust, and Golang

**Most notable programming language**

Rust is among the most notable programming languages, which makes it more notable, also more and more companies are starting to use it, showing its potential.

*Hundreds of companies around the world are using Rust in production today for fast, low-resource, cross-platform solutions. Software you know and love, like*[*Firefox*](https://hacks.mozilla.org/2017/08/inside-a-super-fast-css-engine-quantum-css-aka-stylo/)*,*[*Dropbox*](https://blogs.dropbox.com/tech/2016/06/lossless-compression-with-brotli/)*, and*[*Cloudflare*](https://blog.cloudflare.com/cloudflare-workers-as-a-serverless-rust-platform/)*, uses Rust.****From startups to large corporations, from embedded devices to scalable web services, Rust is a great fit.***



Source: [bestcolleges](https://www.bestcolleges.com/bootcamps/guides/how-many-coding-languages-are-there/" \l ":~:text=According%20to%20the%20Online%20Historical,in%20the%20commonly%20used%20group." \t "_blank)

**Why Are There So Many?**

Coders use different languages for different purposes. Some are used predominantly for web development. Others are used when writing desktop software, [creating mobile apps](https://www.bestcolleges.com/bootcamps/guides/how-to-code-an-app/), and solving scientific problems. Some languages are used for client-side programming, while others are designed for the back-end or server-side.

**Rust and its three key pillars**

If we check the [Rust website](https://www.rust-lang.org/) we can identify that Rust has focused on three key qualities: “Performance,” “Reliability,” and “Productivity.” These qualities are interrelated and contribute to making Rust a versatile and powerful programming language.

**Performance:**

* **Low-level Control:** Rust provides low-level control over system resources without sacrificing safety. It allows developers to manage memory directly, which is crucial for achieving high performance in systems programming.
* **Zero-cost Abstractions:** Rust offers high-level abstractions without incurring runtime overhead. This means that developers can use convenient and expressive programming constructs without sacrificing runtime efficiency.
* **Predictable Performance:**Rust’s ownership system and strict borrowing rules contribute to predictable performance, as they prevent common programming errors that could lead to unpredictable behavior or performance bottlenecks.

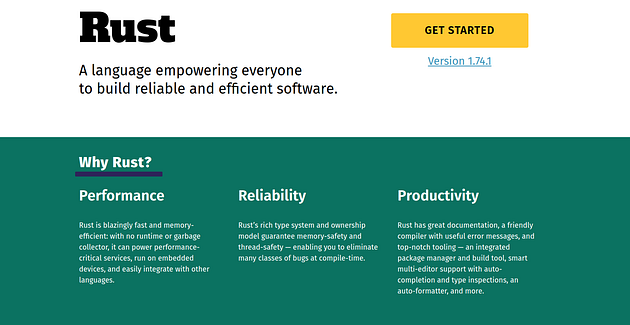
**Reliability:**

* **Memory Safety:**One of Rust’s most distinctive features is its ownership system, which ensures memory safety without the need for garbage collection. This prevents common issues like null pointer dereferences, dangling pointers, and data races.
* **Concurrency without Data Races:**Rust’s ownership system also facilitates safe concurrent programming by preventing data races. The borrow checker ensures that multiple parts of a program cannot simultaneously modify shared data, reducing the likelihood of bugs related to concurrency.
* **Error Handling:** Rust has a robust and expressive error-handling system that encourages developers to handle errors systematically. This helps improve the reliability of Rust's code by making it more resilient to failures.

**Productivity:**

* **Expressive Syntax:** Rust has a clean and expressive syntax that allows developers to write concise and readable code. It incorporates modern programming language features while maintaining a focus on performance and reliability.
* **Cargo Package Manager**: Rust comes with the Cargo Package Manager, which simplifies the process of managing dependencies, building projects, and running tests. This contributes to the overall productivity of Rust developers.
* **Tooling Support:** Rust has a growing ecosystem of tools and libraries that enhance productivity. The language is well-supported by integrated development environments (IDEs) and has a vibrant community that actively contributes to its ecosystem.

By combining these three qualities, Rust aims to provide a programming language that is well-suited for systems programming, embedded systems, and other performance-critical applications while minimizing common programming pitfalls and errors.***This makes Rust an attractive choice for developers working on projects where performance, reliability, and productivity are essential.***



<https://www.rust-lang.org/>

Rust is similar to C and C ++ and other languages that are compiled for machine code. When we write a Rust code this code passes for a compiler and we have a binary as the output, which is an executable file.

**Rust’s History**



Rust began as a side project of Graydon Hoare, an employee at Mozilla. In short order, Mozilla saw the potential of the new language and began sponsoring it, before revealing it to the world in 2010.

One possible source of the name, according to Hoare, is the rust fungus. This has caused Rust programmers to adopt “Rustaceans” as their moniker of choice.

Despite its relative youth, Rust has steadily risen in the ranks of popular programming languages. While it [ranked](https://www.tiobe.com/tiobe-index/) 33 in July 2019, by July 2020 it had risen to the 18th spot on the TIOBE Programming Community Index. Similarly, according to the [Stack Overflow Developer Survey](https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved), Rust has been the “most loved” language since 2016.

**Rust’s Features**

From the beginning, Rust has focused on safety and performance. One of the primary ways it achieves this is by emphasizing safe concurrency. Concurrent computing allows different parts of a program to be executed out of order. This opens up the possibility of parallel computing, where multiple tasks can be accomplished concurrently, rather than consecutively. This, in turn, can significantly improve the performance of an application. Since Rust was designed around this principle, it has some significant advantages in the realm of performance.

In terms of syntax, Rust is similar to C and C++, incorporating many of the keywords and commands from both languages. It is not a direct clone, however, and has some elements not found in either C or C++.

Rust does not have automatic garbage collection, nor does it use Automatic Reference Counting like Swift. Instead, it is designed to be memory-safe by using principles of ownership and borrowing, as well as discouraging undefined behavior. Because it does not use garbage collection, this makes Rust an ideal option for integration with C.

Many of these advantages were touted in the initial blog announcing the Rust 1.0 release:

*“Today we are very proud to announce the 1.0 release of Rust, a new programming language aiming to make it easier to build reliable, efficient systems. Rust combines low-level control over performance with high-level convenience and safety guarantees. Better yet, it achieves these goals without requiring a garbage collector or runtime, making it possible to use Rust libraries as a “drop-in replacement” for C. If you’d like to experiment with Rust, the “Getting Started” section of the Rust book is your best bet (if you prefer to use an e-reader, Pascal Hertleif maintains unofficial e-book versions as well).*

*“What makes Rust different from other languages is its type system, which represents a refinement and codification of “best practices” that have been hammered out by generations of C and C++ programmers. As such, Rust has something to offer for both experienced systems programmers and newcomers alike: experienced programmers will find they save time they would have spent debugging, whereas newcomers can write low-level code without worrying about minor mistakes leading to mysterious crashes.”*

These features have helped Rust make its way into some of the biggest companies in tech, including Firefox, Cloudflare, Dropbox, Yelp, and others.

**Rust is a modern language**

Much of programming language design today focuses on creating functional languages that guide the coder into writing software that’s easier to analyze. Rust is part of this trend. Many developers love Rust’s logical, functional syntax that encourages structuring their code as a sequence of nested function calls.

At the same time, Rust’s creators wanted to build something that could handle the bit-banging, low-level programming required to keep IoT (Internet of Things) functioning. Rust offers the right combination for programmers looking to tackle these very real challenges with modern style.

**Rust has a steep learning curve**

In some regards, learning Rust is a process of unlearning concepts and techniques you’ve likely followed from the beginning of your programming career. As an example, Rust requires abandoning the ideas of scope and ownership, which are required by older languages like JavaScript and Java.

If you want to leverage Rust’s benefits, you have to be willing to relinquish some familiar features that can lead to bugs. Rust’s language syntax is also complex — some would say too much so. It’s not just curly brackets and parentheses, anymore; square brackets, vertical lines, and the greater-than symbol all make an appearance. Sometimes, there are even double colons because one colon isn’t enough.

Rust developers who are building complex, multithreaded tools may see Rust’s syntactic complexity as a worthwhile tradeoff. True fans who grok the functional flow may even enjoy it. Others will just be frustrated. Learning all of Rust’s semantic rules is not for the casual user.

**Rust is not strictly object-oriented**

Rust does not adhere to object-oriented programming principles, which is an issue for some programmers. You can mimic some aspects of object-oriented programming in Rust — true fans know all the best ways to imitate OOP with Rust constructs — but anyone who wants to build elaborate type hierarchies will be frustrated by Rust.

**Rust has great backward compatibility**

Rust’s development team is committed to ensuring that code continues to run even as the language evolves. The team works to ensure that older code continues to compile and run with newer versions of the language, which is something that other languages sometimes ignore. Rust enthusiasts often note that they’re able to maintain their codebase without endless rewriting. That’s because Rust is a language that respects its history.

**Rust’s asynchronous processing model is safer**

Rust’s asynchronous programming model lets developers create separate functions that run independently and then join the results. Many developers say that this structure helps them build faster code while experiencing fewer bugs.

**Asynchronous coding is hard**

Nothing Rust does can save us from having to think carefully about our code. Rust can’t protect the code from deadlocks or delays; it can only offer better advice and a less buggy structure. Developers are still responsible for good application design and writing clean code. It would be nice if Rust were a magic wand, but it isn’t. Rust just minimizes problems and reduces some of the more obvious dangers.

**Programming without abstractions**

Rust was built to support system-level programmers who write low-level, byte-tweaking code. It offers access to the raw bits and expects programmers to use it. The language is designed to cohabitate with much of the old C or assembly language code that’s part of the lower levels of operating systems and network stacks. Real programmers want that access to build the best, most responsive stack. Rust delivers.

**Byte-level access is risky**

Many languages have evolved to avoid byte-level access for a good reason: it’s an easy way for programmers to get into trouble. Hiding the access avoids the risks. Some programmers are better off letting the hidden back end of the language handle the details of allocating memory and representing data.

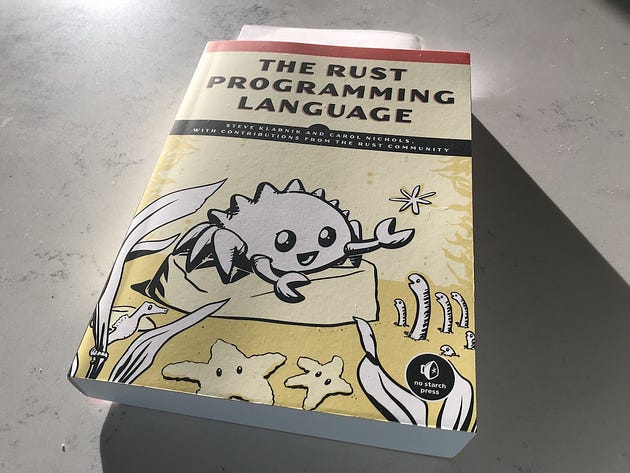
**A better way to do garbage collection**

Many popular languages handle internal memory allocation and garbage collection for you, which is a nice service until the garbage collector stalls everything. It’s bad enough when garbage collection stalls the movie playing on your computer on Friday night. It’s potentially deadly if it strikes a medical device.

The Rust language has an approach to memory management that’s not as comprehensive as traditional GC but can be more powerful. A good developer can deliver great performance using Rust’s memory model, but they’ve got to master the type system and atomic reference counting.

For die-hard Rust fans, hands-on memory management is a feature they love. Even if the job means juggling numerous threads and ensuring the code is responsive, they’d rather do it themselves. For better or worse, Rust puts the power in your hands.

**Rust Projects with Source Code**



**Rust implementation of the cd command**

* Project title: Rust implementation of cd command
* Short description: cd command is used to change present working directory (pwd) to a specified directory path. It is implemented in C on Linux systems. Project is to replace it with a Rust alternative.
* Innovation: New Rust cd command can learn frequently used paths and enable shortcuts.
* Implementation Difficulty: 6 out of 10
* Audience interest: 8 out of 10
* Domain knowledge: Linux, Operating System, GNU coreutils
* [Source code](https://github.com/ajeetdsouza/zoxide/)

**Safe alternative to rm command**

* Project title: Safe alternative to rm command
* Short description: On UNIX systems, rm command deletes the file permanently. With inspiration from Windows, the project is to implement a safe alternative to rm in Rust which move files to a recycle bin directory.
* Innovation: Feature of recycle bin added.
* Implementation Difficulty: 6 out of 10
* Audience interest: 8 out of 10
* Domain knowledge: Linux, Operating System, GNU coreutils
* [Source code](https://github.com/nivekuil/rip)

**Chat server in Rust**

* Project title: Chat server in Rust
* Short description: This is a common project idea but implementing it in Rust brings out the strong components of Rust and hence, is a good project to demonstrate Rust.
* Innovation: Create a chat server that stores messages in the past 10 minutes only.
* Implementation Difficulty: 7 out of 10
* Audience interest: 6 out of 10
* Domain knowledge: Concurrency, System Design
* [Source code](https://github.com/jean553/rust-chat-server)

**Twitter bot in Rust**

* Project title: Twitter bot in Rust
* Short description: The project is to develop a bot to tweet, retweet and like tweets on Twitter.
* Innovation: Find innovative way to analyze data.
* Implementation Difficulty: 5 out of 10
* Audience interest: 7 out of 10
* Domain knowledge: Web API
* [Source code](https://github.com/juli1/twitter-bot-rs)

**Rust for Bioinformatics**

* Project title: Rust for Bioinformatics
* Short description: Scientists are [moving to Rust](https://www.nature.com/articles/d41586-020-03382-2) for analyyzing large amounts of scientific data for various reasons such as speed. Preparing or analyzing a scientific problem using Rust Programming Language will be a good project.
* Innovation: The scientific problem you analyze
* Implementation Difficulty: 9 out of 10s
* Audience interest: 8 out of 10
* Domain knowledge: A scientific domain like Bioinformatics
* [Source code](https://github.com/rust-bio/rust-bio)

**Database in Rust**

* Project title: Database in Rust
* Short description: Creating a database in Rust is a good project as database needs to be secure and performant.
* Innovation: Set specific performance goals
* Implementation Difficulty: 9 out of 10
* Audience interest: 7 out of 10
* Domain knowledge: Database, Cyber-security, Scalability
* [Source code](https://github.com/erikgrinaker/toydb)

**Further Reading:**

* T[alentopia](https://www.talentopia.com/news/the-rust-programming-language-its-history-and-why)
* [Rust Jobs](https://rustjobs.dev/)
* [Rust and the Future of Systems Programming](https://medium.com/mozilla-tech/rust-and-the-future-of-systems-programming-b75fba746910)
* I[nfoworld](https://www.infoworld.com/article/3675391/7-reasons-to-love-the-rust-language-and-7-reasons-not-to.html) [7 reasons to love the](https://www.infoworld.com/article/3675391/7-reasons-to-love-the-rust-language-and-7-reasons-not-to.html)Rust Language
* [IQ opengenus](https://iq.opengenus.org/rust-projects/) [32 Rust Projects](https://iq.opengenus.org/rust-projects/)