

Forum:[Hack Forums](#)

Thread: [Rust-Lang](#)

jasonnix - June 19, 2023, 6:03 am

Hello,

I have started learning Rust language and I want to know is there anyone here who knows this language? If yes, please share you experiences with it.

Is it hard to learn?

Is it good for the future?

Thank you.

MavenGradle - June 20, 2023, 6:49 am

A language being hard to learn depends entirely on the individual who's learning the language due to the word hard being a subjective term. In a general sense, most people will say that Rust is harder to learn than for example Python, but others might not agree. I find Rust easier than JavaScript for example for a myriad of reasons and a lot of people will generally disagree with my statement.

That being said, due to the recent debacles with the Rust Foundation, I'm unsure of Rust's future. It might course-correct, but in general, Rust is not a commonly used language for enterprise-level applications because of the lack of well defined standards (e.g. governance), licensing agreements and generally speaking a lack of talent in the field of Rust programming. What I mean by saying there's a lack of talent when it comes to programming in Rust is that it is much easier to find a Python programmer than it is to find a Rust programmer because there aren't that many Rust programmers out there, statistically speaking, when compared to other programming languages.

In terms of the effectiveness of the Rust programming language, I quite enjoy working with the language, especially because it lacks a garbage collector like in Go or Java's case, but then again, the compilation times for larger applications can really get on my nerves. This is why I recommend having a build server for large-scale Rust applications so that you can spend your time doing productive things while the build server is busy compiling whatever commit/tag/pr/ is currently relevant.

Rust is probably future-proof in the sense that more and more people are starting to learn and utilize Rust (you, being the prime example of that), but due to the recent Rust Foundation debacles, it is once again quite difficult to say what the future has in store for Rust. It might self-destruct, it might flourish and become widely used. It might even be replaced entirely by the next programming language hype (Zig?). Regardless, it is important to utilize the tools and languages that you find are helping you achieve your goals. If your goal is to write efficient and neatly performing code, Rust is probably a good language for you. You can even write an entire website in Rust (compiled to WASM) if you wanted to, so there's really a huge array of use cases for Rust, it's just very sad to see that there isn't a whole lot of actual standards and guidelines currently available. For example, some of the functions in the std are reliant on nightly features, and some features in Rust like working with generics are lacking (async trait) but are being worked on. This is currently hindering the ability to write clean and testable code, but other than that, the question some corporate entities rightfully ask is if Rust code that works and compiles today will still work and compile 10 years from now and if the code written using today's best practices will still be considered 'good' 10 years from

now due to new features being introduced like async traits for example (and if the required talent will be available in said future to actually do some work). After all, imagine someone at a company writing a super-complex and advanced Rust server application that is used by thousands of people (not even that big of a corporation), then leaving after some time and having to find a replacement for when things eventually stop working due to no particular reason other than the evolution of operating systems, software, CPU architectures and more. Would this company then be forced to rewrite the entire thing from scratch because of old and obsolete methods being used (even though they were once available in the std)? I don't think a company likes the idea of uncertainty when it comes to the applications they themselves run. Especially retail companies. They just want things to work and continue working for a long time, generally speaking, without having to invest again and again in writing the best possible code. This is also partially why ERP systems like SAP and in general why software houses are a thing: to abstract the cost of software (and its maintenance) away from the retailer, in the hopes that the contracted party maintain things for them. Of course, software vendors promise a lot of things. Statements like 'it just works' and 'you will never need to worry about it again' are quite common sale tactics and companies are sooner or later forced to do some kind of 'upgrade' anyway at some point down the line. Otherwise the entities maintaining the systems would run out of cash or talent quite soon (who wants to be the legacy maintainer when more new and exciting things can be made). That being said, there are certainly corporate entities that have bitten the Rust bullet and are currently using Rust for production purposes in large-scale applications and they probably have some well-paying positions open. It might however be quite difficult to find job positions for Rust programming in your area, it all depends. Enough about business, I digress. Apologies.

In summary, the Rust programming language itself is fine, it's just missing some features that are being worked on, but it still offers a wide range of possibilities and use-cases today that make it a viable language for projects of any size, as long as you find the resources and willingness to work with the programming language and are willing to overlook some of the quirks of the Rust foundation. This is only relevant for today, the date this comment was made, and might not apply next year if some absurd political decision is made by the foundation (and it makes me quite sad to have to write this out as well, because last year I would have said everyone should learn Rust with zero doubts in my mind, but unfortunately the Rust Foundation has been quite a pain in the ass for the Rust language itself and the reputation it holds. This entire statement shouldn't even be a thing that needs to be written out but alas, here we are in 2023).

Here's some relevant reading material to check out in case you're bored or if you want to do some fact-checking:

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

<https://blog.rust-lang.org/inside-rust/2023/05/03/stabilizing-async-fn-in-trait.html>

<https://blog.rust-lang.org/inside-rust/2023/04/12/trademark-policy-draft-feedback.html>

<https://analyticsindiamag.com/internal-drama-unveiled-a-behind-the-scenes-look-at-rust/>

<https://www.youtube.com/watch?v=9z82HO31mxU&themeRefresh=1>

Sajuuk - June 21, 2023, 7:51 pm

(Jun 20, 2023 06:49 AM)MavenGradle Wrote: [\[CITE\]](#) -all that-

Giving thorough and detailed responses on HF be like: [\(Click to View\)](#)

jasonnix - June 28, 2023, 4:03 am

(Jun 20, 2023 06:49 AM)MavenGradle Wrote: [\[CITE\]](#) A language being hard to learn depends entirely on the individual who's learning the language due to the word hard being a subjective term. In a general sense, most people will say that Rust is harder to learn than for example Python, but others might not agree. I find Rust easier than JavaScript for example for a myriad of reasons and a lot of people will generally disagree with my statement.

That being said, due to the recent debacles with the Rust Foundation, I'm unsure of Rust's future. It might course-correct, but in general, Rust is not a commonly used language for enterprise-level applications because of the lack of well defined standards (e.g. governance), licensing agreements and generally speaking a lack of talent in the field of Rust programming. What I mean by saying there's a lack of talent when it comes to programming in Rust is that it is much easier to find a Python programmer than it is to find a Rust programmer because there aren't that many Rust programmers out there, statistically speaking, when compared to other programming languages.

In terms of the effectiveness of the Rust programming language, I quite enjoy working with the language, especially because it lacks a garbage collector like in Go or Java's case, but then again, the compilation times for larger applications can really get on my nerves. This is why I recommend having a build server for large-scale Rust applications so that you can spend your time doing productive things while the build server is busy compiling whatever commit/tag/pr/ is currently relevant.

Rust is probably future-proof in the sense that more and more people are starting to learn and utilize Rust (you, being the prime example of that), but due to the recent Rust Foundation debacles, it is once again quite difficult to say what the future has in store for Rust. It might self-destruct, it might flourish and become widely used. It might even be replaced entirely by the next programming language hype (Zig?). Regardless, it is important to utilize the tools and languages that you find are helping you achieve your goals. If your goal is to write efficient and neatly performing code, Rust is probably a good language for you. You can even write an entire website in Rust (compiled to WASM) if you wanted to, so there's really a huge array of use cases for Rust, it's just very sad to see that there isn't a whole lot of actual standards and guidelines currently available. For example, some of the functions in the std are reliant on nightly features, and some features in Rust like working with generics are lacking (async trait) but are being worked on. This is currently hindering the ability to write clean and testable code, but other than that, the question some corporate entities rightfully ask is if Rust code that works and compiles today will still work and compile 10 years from now and if the code written using today's best practices will still be considered 'good' 10 years from now due to new features being introduced like async traits for example (and if the required talent will be available in said future to actually do some work). After all, imagine someone at a company writing a super-complex and advanced Rust server application that is used by thousands of people (not even that big of a corporation), then leaving after some time and having to find a replacement for when things eventually stop working due to no particular reason other than the evolution of operating systems, software, CPU architectures and more. Would this company then be forced to rewrite the entire thing from scratch because of old and obsolete methods being used (even though they were once available in the std)? I don't think a company likes the idea of uncertainty when it comes to the applications they themselves run. Especially retail companies. They just want things to work and continue working for a long time, generally speaking, without having to invest again and again in writing the best possible code. This is also partially why ERP systems like SAP and in general why software houses are a thing: to abstract the cost of software (and its maintenance) away from the retailer, in the hopes that the contracted party maintain things for them. Of course, software vendors promise a lot of things. Statements like 'it just works' and 'you will never need to worry about it again' are quite common sale tactics and companies are sooner or later forced to do some kind of 'upgrade' anyway at some point down the line. Otherwise the entities maintaining the systems would run out of cash or talent quite soon (who wants to be the legacy maintainer when more new and exciting things can be made). That being said, there are certainly corporate entities that have bitten the Rust bullet and are currently using Rust for production purposes in large-scale applications and they probably have some well-paying positions open. It might however be quite difficult to find job positions for Rust programming in your area, it all depends. Enough about business, I digress. Apologies.

In summary, the Rust programming language itself is fine, it's just missing some features that are

being worked on, but it still offers a wide range of possibilities and use-cases today that make it a viable language for projects of any size, as long as you find the resources and willingness to work with the programming language and are willing to overlook some of the quirks of the Rust foundation. This is only relevant for today, the date this comment was made, and might not apply next year if some absurd political decision is made by the foundation (and it makes me quite sad to have to write this out as well, because last year I would have said everyone should learn Rust with zero doubts in my mind, but unfortunately the Rust Foundation has been quite a pain in the ass for the Rust language itself and the reputation it holds. This entire statement shouldn't even be a thing that needs to be written out but alas, here we are in 2023).

Here's some relevant reading material to check out in case you're bored or if you want to do some fact-checking:

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

<https://blog.rust-lang.org/inside-rust/2023/05/03/stabilizing-async-fn-in-trait.html>

<https://blog.rust-lang.org/inside-rust/2023/04/12/trademark-policy-draft-feedback.html>

<https://analyticsindiamag.com/internal-drama-unveiled-a-behind-the-scenes-look-at-rust/>

<https://www.youtube.com/watch?v=9z82HO31mxU&themeRefresh=1>

Hello,

Thank you so much for your reply.

So, Rust-Lang or CrabLang?

MavenGradle - June 28, 2023, 10:30 am

(Jun 28, 2023 04:03 AM)jasonnix Wrote: [\[CITE\]](#) Hello,

Thank you so much for your reply.

So, Rust-Lang or CrabLang? So you want the TLDR!

TLDR:

For now, Rust. If the Rust Foundation releases a trademark and usage policy that is unacceptable, use Crab (or something else entirely like Zig, C++, etc.). A programming language is just a tool to get shit done at the end of the day. Pick the right one that suits your needs and is acceptable given your views in life (what the fuck, why does this even have to be said in regards to programming language usage - thanks Oracle and Rust Foundation).

DashPHP - June 29, 2023, 12:46 am

Some great responses here so I just wana add that, using something like ChatGPT to help with your learning will be pretty darn useful.