

RUSTIFY

From Java to Rust
The Senior Dev's Guide

For Java devs with 4+ years experience

RUSTIFY

TABLE OF CONTENTS

From Java to Rust

01 You're already halfway there p.3

02 The real cost of staying in Java p.4

03 The engineer you become after Rust p.5

04 The market has a supply problem p.6

05 Is self-teaching Rust the best way? p.7

06 The cost of "not now" p.8

07 "Real Rust devs come from C/C++" p.9

08 Why Rustify p.10

09 FAQ p.11

01

You're already halfway there

The transition feels steep from the outside. From the inside, it's faster than most Java devs expect, because the mental models Java forced you to build transfer directly.

What maps 1:1

```
// Java                          // Rust
interface Drawable { ... }   →   trait Drawable { ... }
Optional<T>                  →   Option<T>
throws IOException           →   Result<T, io::Error>
List<T>, Map<K,V>            →   Vec<T>, HashMap<K,V>
generics <T extends X>       →   generics <T: X>
final                        →   let (immutable by default)
```

You already think in types, generics, and interfaces. Rust just calls them different names and makes them more powerful.

What your Java experience buys you

You understand compile-time guarantees. You've written `Optional.ofNullable` to avoid NPE. You know why immutability matters. Rust takes all of that and enforces it at a level Java never could.

🎯 10 years of Java didn't disappear. It became the foundation Rust builds on.

02

The real cost of staying in Java

In 2015, knowing Java well was a differentiator. Spring Boot was new, senior Java devs were scarce, and the market rewarded expertise with leverage. That's not the market anymore.

You'll always be good. You might always be replaceable.

Java has been around long enough that there are hundreds of thousands of competent Java devs. Senior ones. With 10 years of Spring Boot. When you apply for a role, that's your competition pool. Being good isn't the differentiator anymore. Being rare is.

Salary progression plateaus

Most senior Java devs hit a ceiling between year 5 and year 8. The next level isn't about being a better Java dev. It's about being valuable in a way that fewer people can replicate. Java doesn't give you that leverage anymore.

AI is doing the boilerplate

The Spring Boot controller. The Hibernate mapping. The DTO factory. The parts of Java that took experience to write well are the parts AI handles in seconds now. What's left is judgment and architecture. That's valuable anywhere, but more valuable in a stack where fewer people can even read the code.

> 🎯 You're not deciding between Java and Rust. You're deciding between being one of many and being one of few.

03

The engineer you become after Rust

The fear most senior Java devs have is this: "I'll lose my seniority. I'll go back to being a junior." That's not what happens.

What actually happens is that you bring 10 years of systems thinking, API design, debugging instincts, and delivery experience, applied in a language that removes entire categories of runtime failures. You don't start over. You compound.

What senior Java devs report after 6 months in Rust

- Debugging sessions are shorter. The compiler catches what used to surface in prod.
- They write less defensive code. No null checks, no try/catch boilerplate.
- They feel "senior again", in a space where the bar is still being set.
- Hiring managers treat them differently. A Java dev with Rust is rare. A senior one is very rare.

🎯 The Rust job market is still early. Senior Java devs who cross over now are entering a space where seniority is genuinely scarce. That's not a risk. That's timing.

04

The market has a supply problem

There are more Rust job openings than qualified developers. That's not a prediction. It's the current state. Companies aren't waiting for the perfect Rust dev. They're hiring the best available one.

You don't need to be exceptional. You need to exist in a market where most engineers haven't arrived yet.

Where the demand actually is

- **Fintech & trading** Performance-critical backends where GC pauses are unacceptable
- **Infrastructure & DevTools** CLIs, build systems, platform tooling
- **B2B SaaS scale-ups** Rewriting bottlenecks from Python/Go/Java as they scale
- **Web3 & embedded** Systems work that previously required C/C++

The hiring managers posting these roles aren't finding who they need. Most Rust job openings sit open for 3–6 months. When a senior backend engineer with Java experience shows up and can talk Rust, the conversation is different from the start.

🎯 The supply problem works in your favor. You don't need to outcompete hundreds of candidates. You need to be one of the few who showed up.

06

Is self-teaching Rust the best way?

When async or multithreading stopped making sense: who did you call?

If the answer is "no one", that's the data. Async Rust and ownership across threads don't just need time. They need someone who's already restructured their thinking around it, who can show you why your Java mental model doesn't map, and what replaces it. Stack Overflow threads don't do that.

Do you know which crates you'd use for your first production API?

Which middleware pattern for auth in Axum? Which sqlx pattern for a transaction that touches 3 tables? This isn't in any tutorial. It's assembled over months, mostly wrong at first, unless someone shows you the short path.

How long before you'd feel confident shipping?

Most engineers who go alone give an honest answer when pressed: "probably a year, maybe more." That's not a guess. It's what the path actually looks like without structure.

🎯 Self-teaching Rust is possible. The question is whether your goal is to learn Rust eventually, or to ship it within months.

07

The cost of "not now"

You have a full-time job. Maybe a partner, maybe kids. Real constraints, not excuses. This isn't about pretending otherwise.

But "no time" and "this isn't a priority right now" are different things. And knowing which one is true changes everything.

What specifically needs to change before you'd have time?

A project wraps up. A release ships. Things calm down. When did they last calm down, and what did you do with that time? The engineers who are shipping Rust today didn't wait for a quiet period. They decided it was a priority.

What would you have today if you'd started 6 months ago?

A deployed Axum service. A credible Rust portfolio. A different set of conversations with hiring managers. Six months from now, this question has the same answer, unless something changes.

Is it time, or is it priority?

If you genuinely have no bandwidth right now, that's a real answer. But if the honest answer is "I'm not sure it's worth it": that's a different question. And it's the one this guide is trying to help you answer.

> 🎯 The engineers who say "I'll do it when things calm down" are still waiting. The ones who shipped are the ones who decided before things calmed down.

07

"Real Rust devs come from C/C++"

That was true in 2017. Today, most new Rust developers come from TypeScript, Python, and Java. The systems crowd opened the door. The backend world walked through it.

Who's actually building in Rust today

The majority of Rust web and backend developers come from TypeScript, Python, and Java. Not C. The systems programmers were the early adopters. The backend engineers are the current wave. That's the community you're entering.

You'll be ahead, not behind

A Java dev who transitions has more production backend experience than most people who picked up Rust in the last two years. You know what it means to debug a service in prod, design an API for scale, and work in a team codebase. Most early Rust devs are still learning that.

The question isn't where you come from

The Rust community doesn't ask about your background. It asks what you can build. Two years from now, you'll be the senior Rust dev that new people look to for guidance. The impostor is the one who never starts.

> 🎯 The people who look down on Java devs learning Rust are the ones who haven't shipped anything. The ones who have, are too busy to care.

RUSTIFY

SUCCESS STORIES

Engineers who made the switch



Ugo Tiberto

Rust Engineer · Fullstack

Ugo started with JavaScript 3 years ago, then decided to push further. He joined the bootcamp because learning Rust basics alone is easy, but building real apps with Axum, Leptos, and SQLx requires structure. Two weeks after finishing, he had a production Leptos app deployed.

Eddy Kamto

Software Engineer · Rust / TS

Eddy is a fullstack engineer with 5 years of TypeScript. He joined after seeing Rust UI, a Shadcn-like component library built entirely in Rust. He was looking for a structured path through a language with a steep curve and a vast ecosystem.



Tiago Afonso

Fullstack Developer



Tiago studied computer science in India and graduated in 2024. He first encountered Rust through Bitcoin infrastructure work. He joined the bootcamp after seeing a LinkedIn post about using Rust for full-stack web development, not just systems programming.

09

FAQ

Q1. I tried learning Rust on my own and got stuck on the borrow checker. Is that on me?

Getting stuck solo is almost universal. It's not a signal about your ceiling. It's a signal about your context. The borrow checker needs feedback: someone who shows you not just what the error means, but how to restructure the thinking behind the code. Self-teaching works for syntax. It breaks down here.

Q2. Can't I just learn this on my own? Free content is everywhere.

You can learn the concepts for free. The question is how long before you're shipping production Axum APIs with confidence. Most engineers who go alone take 12–18 months, if they don't quit first. The bottleneck isn't information. It's a structured path, someone to unblock you at the right moments, and accountability through the hard weeks.

Q3. My company runs Spring Boot. Is Rust practical for microservices?

Yes. Axum is production-ready and used at scale. You don't migrate everything at once. Write one new service in Rust, benchmark it, and let the results make the case. Most teams start with a high-traffic or latency-sensitive service.

Q4. How long until I'm productive? I can't afford to be a junior again.

Senior Java devs typically ship their first Rust API in 6–8 weeks. Not a toy. An actual deployed service. The bottleneck is ownership, not the rest of the language. Once that clicks, your Java experience accelerates everything else.

Q5. Is the ecosystem mature enough for production backend work?

For backend: yes. `axum`, `sqlx`, `tokio`, `serde` are stable, widely used, and actively maintained. The ecosystem is narrower than Java's but the quality-to-choice ratio is higher. You won't miss Spring Boot's 47 configuration layers.

Targeting a Rust role? **Check if you're eligible for Rustify coaching →**



Ready to make the move?

Your Java experience is an asset, not a sunk cost.
We work with a small number of senior engineers.
Check if you qualify.

Check my eligibility →