About the course

In PHP

You already ship software. PHP runs the request, Composer brings in the libraries, and you find out about a bad value when that line actually runs. None of that is wasted. This course adds Rust beside that, for the CLI, the worker, or the small service where you want the compiler to refuse a whole class of bugs before the binary exists.

Rust fits next to PHP for CLIs, queue workers, small high-throughput services, and WebAssembly. The main PHP app stays as it is.

In Rust

Rust is a compiled language. `rustc` turns your source into a native binary, and that binary is what you run. There is no garbage collector. The compiler tracks who owns each value, and who is only borrowing it, and it will not produce the binary if those rules are broken.

You do not need to throw away PHP to learn that. The useful habit is the same one you already have: name the data, pass it into a function, and handle the failure. Rust spells the types in the signature, and it checks them when you compile, not when a rare request finally hits that line.

By the end of this chapter you can:

- install Rust and confirm `rustc` and `cargo`
- tell a compile-time type error from PHP quietly coercing `"5"` into an int
- create a project with Cargo the way you create one with Composer
- pin the toolchain version for the team in `rust-toolchain.toml`
- write a tiny calculator that stops at end of input, and run `cargo test`

Libraries (`cargo new --lib` , `src/lib.rs`) wait for the modules chapter. Ownership waits for its own chapter. You will see `Option` once, in the calculator, with a note that the real lesson comes later.

How this course works

Each lesson is built the same way:

- **In PHP** — the behaviour you already know
- **In Rust** — the same idea in Rust
- **Side by side** — a short table
- **Try it** — something to run in the Playground or locally

At the end of a chapter, a quiz checks the chapter. You need **80%** to pass.

Use the **Playground** button on a lesson to open that lesson's Rust in the official sandbox. The GitHub examples repo holds example crates. Lessons that have one show a GitHub button. **Quick reference** is the PHP → Rust cheat sheet (open it from the course page or a lesson). **PDF downloads** give you a printable copy of a lesson or the whole chapter.

Prerequisites: comfort with PHP (functions, Composer, running a CLI script). No prior Rust. Roughly **1–2 hours per chapter** if you run the Try it steps.

Key takeaways

- Rust sits next to PHP. It does not replace the apps you already run.
- The compiler checks types and ownership before you have a binary.
- This chapter is install, Cargo, pinning, and one small CLI.

What's next: What Rust checks for you