What you learned

In PHP

You install PHP and Composer, you run source files directly, and a typed function will coerce `"5"` unless that file says `strict_types`. Composer's lockfile is how an app builds the same way twice. Nothing in this chapter asked you to stop doing that.

In Rust

This chapter taught five things:

1. **What Rust checks** — a type mismatch (`E0308`) or a live conflicting borrow (`E0502`) stops the compile; PHP often coerces until that line runs.
2. **Install with rustup** — one installer for `rustc` and `cargo`, confirm with `rustup show`.
3. **Cargo** — `cargo new`, `cargo run`, `cargo fmt`, `cargo clippy`, debug vs release (including overflow behaviour).
4. **Dependencies and pinning** — `cargo add`, commit `Cargo.lock` for apps, pin the team toolchain in `rust-toolchain.toml` (never `rustup default nightly`).
5. **A small CLI** — read lines, stop at EOF, test pure arithmetic with `cargo test`.

`[dev-dependencies]` is Composer's `require-dev`. CI should pass `--locked`. The site does not run those tests for you. Run them locally, or use the Playground **Test** action on the calculator lesson.

Key takeaways

1. rustup installs the toolchain; pin a specific stable version for the team.
2. Cargo is the everyday tool: new, run, fmt, clippy, add, test.
3. PHP coerces by default. Rust does not compile the mismatch.
4. `Cargo.lock` and `--locked` keep builds repeatable.

5. The calculator stops at EOF, and `cargo test` covers the arithmetic.

Take the chapter quiz from the button on this page. It opens a separate page. Thirteen questions, and you need 80% to pass. After you submit, each question shows a one-line explanation.

What's next: Variables