Hello Rust

Getting started · Hello Rust

In PHP

`php hello.php` reads the file and runs it. There is no separate binary. Whatever `echo` prints is the program's output. OPcache can cache the opcodes so PHP does not re-parse every request, but you still ship and run source — there is no standalone binary sitting next to the file.

hello.php

```
<?php

echo "Hello, Rust!\n";
```

In Rust

A Rust program needs a `main` function. `println!` writes a line to stdout. The `!` means it is a macro, not a function. You can ignore that until macros have their own moment. The useful fact is that this file is not run directly.

hello.rs

```
fn main() {
    println!("Hello, Rust!");
}
```

```
rustc --edition 2024 hello.rs
./hello
```

On Windows the binary is `hello.exe`. `rustc` writes the executable next to the source file. That file is what you run.

The binary carries the standard library compiled in. There is no separate Rust runtime or interpreter to install on the machine that runs it. It only runs on the same OS and CPU target you compiled for (a Linux `x86_64` binary will not run on macOS ARM, and so on). Cross-compiling is a later topic; for this course, build on the machine where you will run.

`--edition 2024` matters when you invoke `rustc` yourself. Cargo puts the edition in `Cargo.toml` so you do not pass the flag each time.

> **Watch for:** treating `rustc hello.rs && ./hello` as the normal workflow. It works for one file. The moment you want dependencies, tests, or fmt, you want Cargo.

> **Idiomatic Rust:** `fn main()` is the entry point of a binary. There is no class and no framework.

Side by side

	PHP	Rust
Run the file	`php hello.php`	`rustc` writes a binary, then you run that
The process	The interpreter reads the source again	The binary is the program you run
Next step	Stay with the interpreter	`cargo` once you want dependencies or tests

Try it

Compile the file and run it. Then change the line to `println!("Hello, {name}!");` and compile. `rustc` reports `E0425` (cannot find value `name`). Then add `let name = "Rust";` above it and compile again.

Key takeaways

- `rustc` produces a binary. PHP runs the source.
- Use Cargo for anything beyond a single file.

What's next: What is Cargo?