Rust Programming Guidebook

What is a Rust "hello world" program?

What is a Rust "FizzBuzz" program?

What makes Rust good?

Is programming language theory in Rust?

What is the Rust ecosystem?

Who are Rust leaders?

Who might benefit from learning Rust?

What are good ways to learn Rust?

What are good projects to learn Rust?

What are the hardest parts of Rust?

What is Rust missing?

Why do companies not use Rust?

Borrow checker

Channels for thread communication

Concurrency and parallelism

Error messages

Foreign Function Interface (FFI)

Futures for asynchronous operations

Monomorphisation

Rust stable versus Rust nightly

Unsafe code

WebAssembly (WASM)

Zero-cost abstractions

Rust versus C/C++

Rust versus Go

Rust versus Java

Rust versus JavaScript

Rust versus Nim

Rust versus Python

Rust versus Zig

Rust for artificial intelligence

Rust for financial technology (fintech)

Rust for embedded devices

Rust for game development

Rust for graphical user interfaces (GUIs)

Rust for Linux drivers

Scalar types

Compound types

Tuples for ordered collections

Box type for a smart pointer

Rc type for single-thread sharing

Arc type for multi-thread sharing

Pin type for memory location

Copy trait and Clone trait for duplicating values

Debug trait for debugging and printing

Display trait for formatting

dyn trait for dynamic dispatch

Eq, PartialEq, Ord, PartialOrd, Hash traits

From and Into traits for conversions

Send and Sync traits for multithreading

async/await keywords for asynchronicity

enum keyword for enumerations

match keyword for control flow

mod keyword for module namespaces

struct keyword for custom data types

trait keyword for polymorphism

catch_unwind! macro for handling panic

macro_rules! macro for declarative macros

Annotations for compiler directives

Destructuring into components

Iterators for traversing collections

Closures for anonymous functions

Macros for metaprogramming

Panic and how to handle it with a hook

Range syntax for a sequence of values

Memory lifetimes

Memory on the stack or the heap

Memory ownership and borrowing

Mutability and immutability

Test framework and test assertions

Unit testing

Integration testing

Documentation testing

Source-based coverage

Test-driven development (TDD)

Access a database with rusqlite

List directories recursively with walkdir

Make HTTP GET request with reqwest

Parse JSON data with Serde

Read a spreadsheet with CSV

Run a terminal program with cursive

Search a text file with regex

rustup command-line tool

Cargo package manager and crates

Clippy linting

Rustfmt for code formatting

Rust mdBook for documentation

Cross-compiling for multiple platforms

Rhai script

Abstract syntax tree (AST)

Tree-sitter parsing library

Language Server Protocol (LSP)

Static analysis for error detection

Design patterns

Dependency injection (DI)

Domain-driven design (DDD)

Model-view-controller (MVC)

Object-oriented versus functional

Procedural versus functional

Resource Acquisition Is Initialization (RAII)

SOLID principles for software design

The Law of Demeter

Assertables crate for assert macro tests

itertools crate for iterator extras

log crate for logging messages

once_cell crate for lazy global variables

regex crate for regular expressions

reqwest crate for HTTP requests

Serde crate for serialize/deserialize

walkdir crate for traversing directories

CLAP crate for command line arg parsing

Cursive crate for text-based interfaces

Textwrap crate for text wrapping

cargo-cache crate for caching builds

cargo-dist crate for distribution archives

cargo-release crate for release automation

cargo-make crate for task runners

Crossbeam crate for concurrency

parking_lot crate for synchronization

Rayon crate for parallelism

arrow-csv crate for loading CSV to Arrow

CSV crate for comma-separated values

Polars crate for data analysis

Diesel crate for object-relational mapping

Rusqlite crate for SQLite databases

sqlx crate for SQL databases

axum crate for web services

Hyper crate for HTTP clients/servers

Tokio crate for asynchronicity/concurrency