

**Rust for JavaScript/TypeScript Developers**

**Course Number:** RUST-114  
**Duration:** 5 days

**Overview**

This Rust training course teaches JavaScript and TypeScript Developers how to write high-performance and secure systems-level software using Rust programming. Attendees learn how to seamlessly transition from JavaScript/TypeScript to Rust, implementing its safety and performance for building next-generation web and server applications.

**Prerequisites**

* Proficiency in Python programming
* Basic understanding of programming concepts such as variables, expressions, functions, and control flow

**Materials**

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub through documentation and extensive code samples.

**Software Needed on Each Student PC**

* A free, personal GitHub account to access the courseware
* Permission to install Rust and Visual Studio Code on their computers
* Permission to install Rust Crates and Visual Studio Extensions

If students cannot configure a local environment, a cloud-based environment can be provided.

**Objectives**

* Understand the Rust philosophy
* Set up and navigate the Rust environment
* Explore Rust within the context of JavaScript/TypeScript
* Understand basic Rust syntax and semantics
* Implement control flow and logic
* Understand ownership and borrowing concepts
* Use tuples, enums, structs, and vectors
* Work with pattern matching
* Understand Rust’s concurrency model
* Connect a Rust application to a database
* Build a client-side WebAssembly app with Leptos
* Build a web API with Actix

**Outline**

* Introduction
* What is Rust?
  + Rust’s Philosophy and Goals
  + History and motivation
  + Rust vs JavaScript
  + Rust vs TypeScript
  + Rust Community
  + The Rust Playground
* Install Rust
  + Script
  + macOS Homebrew
  + Platform Installers
* Rust Editors
  + VSCode with Extensions
  + Rust Rover
  + Debug Rust in VSCode
  + GitHub Copilot
* Hello World
  + Create a new Project
  + Main Function
  + Print to the Console
  + Comments
* Cargo
  + What is Cargo?
  + How does Cargo compare to Pip and Conda?
  + Rust Crates compared to Python Packages
  + Run Command
  + Build Command
  + Build Release Command
  + Install Third-Party Crates
* Rust and JavaScript Differences
  + Static Typing vs Dynamic Typing
  + Strong Typing vs Loose Typing
  + Memory Management
  + Error Handling
  + Sequence, Selection, and Iteration
  + Structs vs Classes & Object Literals
  + Traits vs Duck-Typing
  + Concurrency
* Rust and TypeScript Differences
  + Structural Typing
  + Memory Management
  + Error Handling
  + Sequence, Selection, and Iteration
  + Structs vs Classes & Object Literals
  + Traits vs Interfaces
  + Generics
  + Concurrency
* Scalar Types and Data
  + Rust Types vs Python Type
  + Constants
  + Immutable Variables
  + Mutable Variables
* Code Logic
  + If Statement
  + Loop with Break
  + While Loop
* Functions
  + Define a Function
  + Call a Function
  + Parameter Types
  + Return Types
  + Closure Functions
* Modules
  + Import Modules from Standard Library
  + Import Modules from Third-Party Crates
  + Define Custom Modules
  + Import Custom Modules
* Built-In Macros
  + print! and println!
  + format!
  + vec!
  + include\_str! and include\_bytes!
  + cfg! and env!
  + panic!
* Memory Management
  + Problems with Manual Management
  + Problems with Garbage Collection
  + Ownership & Borrowing
  + Rust vs Python
  + References
  + Lifetimes
* Strings
  + String Slices
  + String Objects
  + Convert Between Slices and Strings
  + Parse Number from String
  + Trim String
  + Print Strings with Interpolation
* Tuples
  + What is a Tuple?
  + Heterogeneous Elements
  + Access Elements
  + Destructuring
  + Immutable
* Enums
  + What is an Enum?
  + Define an Enum
  + Using Enums
  + Enum Variants
  + Enum Methods
  + Enums and Pattern Matching
  + Result Enum
  + Option Enum
  + Enums vs Structs
* Structs
  + What is a Struct?
  + Create Instance
  + Field Init Shorthand
  + Struct Update Syntax
  + Tuple Structs
  + Unit-Like Structs
  + Ownership of Struct Data
  + Function Implementation
  + Associated Functions
  + Stuct Methods
  + Constructor Pattern
* Vectors
  + What is a Vector?
  + Create a Vector
  + Add and Remove Elements
  + Access Elements
  + Iterate over Elements
  + Slicing, Length, and Capacity
  + Common Vector Operations
  + Understand Memory Management
  + Ownership and Borrowing Rules
* Collections and Iterators
  + Vectors, arrays, and slices
  + HashMaps and hash sets
  + Iteration and iterators
* Traits
  + What is a trait?
  + How does a trait related to traditional OOP interfaces?
  + Defining a trait
  + Implementing a trait
  + Default implementations
  + Traits as parameters
  + Traits as return types
  + Traits as bounds
* Generics
  + What is a generic?
  + How does a generic related to traditional OOP generics?
  + Defining a generic
  + Implementing a generic
  + Generic bounds
  + Multiple generic types
  + Where clauses
* Pattern Matching
  + What is Pattern Matching?
  + Match Statement
  + If Let Statement
  + While Let Statement
  + Destructuring Stucts and Tuples
  + Pattern Matching with Enums
  + Pattern Matching with Functions
  + Pattern Matching and Ownership
  + Refutability and Irrefutability
* Concurrent Programming
  + What is Concurrent Programming?
  + Using Multiple Threads
  + Mutex, RwLock, and Arc
  + Message Passing with Channels
  + Sync and Send Traits
  + Futures and Async/Await
* Database Programming
  + What is a Database?
  + Connect to Postgresql
  + Query data from the database
  + Modify data in the database
* Web Assembly with Leptos
  + What is Web Assembly?
  + What is Leptos?
  + How does Leptos compare to React, Angular, and Blazor?
  + Create a Leptos Project
  + Connect it to an Active Web Api
  + Create a Component
  + Pass Data to a Component
  + Emit Events from a Component
  + Work with Signals
  + Render a Collection of Data
* Web APIs with Actix
  + What is a Web API?
  + What is Actix?
  + How does it compare to Express, Flask, and ASP.NET MVC?
  + Create an Actix Project
  + Map HTTP Routes to Rust Functions
  + Working with Extractors
  + Using Handlers
  + Connect to a Database
* Conclusion