Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Rust for JavaScript/TypeScript Programmers

Class Duration

- 5 days of comprehensive training
- Tailored for JavaScript/TypeScript developers

Target Audience

- Proficiency in JavaScript/TypeScript programming
- Basic understanding of programming concepts such as variables, expressions, functions, and control flow

Description

Rust for JavaScript/TypeScript Developers is a dynamic and practical course designed to teach JavaScript/TypeScript developers how to build web and server applications with Rust. Tailored for developers familiar with JavaScript and TypeScript, this course empowers you to harness the unparalleled safety and performance of the Rust programming language. Discover how to leverage your existing web development skills while delving into Rust's robust type system, memory management, and concurrency features. Through hands-on exercises and real-world examples, you'll gain the expertise to confidently develop secure, efficient, and high-performance web and server applications, making this course an invaluable asset for those looking to expand their horizons in the world of programming.

Learning Objectives

- Understand the Rust Philosophy
- Set Up and Navigate the Rust Environment
- Explore Rust within the context of JavaScript/TypeScript
- Grasp Basic Rust Syntax and Semantics
- Learn Control Flow and Logic
- Learn Ownership and Borrowing Concepts
- Utilize Tuples, Enums, Structs, and Vectors
- Employ Pattern Matching
- Harness Rust's Concurrency Model
- Connect a Rust Application a Database
- Build a Client-Side WebAssembly App with Leptos

- Build a Web API with Actix

Training Materials

All students receive comprehensive courseware covering all topics in the course. Courseware is distributed via GitHub in the form of documentation and extensive code samples. Students practice the topics covered through challenging hands-on lab exercises.

Software Requirements

Students will need a free, personal GitHub account to access the courseware. Students will need permission to install Rust and Visual Studio Code on their computers. Also, students will need permission to install Rust Crates and Visual Studio Extensions. Students will need a local instance of Postgresql or SQL Server installed on their computer (using Docker is acceptable). Finally, students will need CURL and/or Postman installed on their computer. If students are unable to configure a local development environment, a cloud-based environment can be provided.

Training Topics

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 NPM and Yarn?
- Rust Crates compared to NPM Packages
- Run Command
- Build Command
- Build Release Command
- Install Third-Party Crates

Popular Cargo Crates

- Serde
- Tokio
- Reqwest
- SQLx
- Anyhow

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 JavaScript/TypeScript Types
- 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!

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Memory Management

- Problems with Manual Management
- Problems with Garbage Collection
- Ownership & Borrowing
- Rust vs JavaScript/TypeScript
- 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
- Struct 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 Structs 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

- Course wrap-up and next steps