

**Desktop Apps with Rust and Tauri**

**Course Number:** RUST-106
**Duration:** 3 days

**Overview**

This Rust training course teaches attendees how to build desktop applications using Rust and Tauri to create UI elements such as windows, menus, and tray icons. Participants learn how to create dialogs, notifications, file system dialogs, and more.

**Note:** This course can be taught with Leptos, React, Angular, Svelte, or Solid.js.

**Prerequisites**

This course assumes prior experience with Rust, JavaScript, HTML, and CSS.

**Materials**

All Rust training 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**

Students will need a free, personal GitHub account to access the courseware. Students 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). If students cannot configure a local development environment on their machines, a cloud-based environment can be provided.

**Objectives**

* Use modern approaches to desktop app development
* Explore how to create desktop apps with Rust and a web browser
* Create a new desktop app using Tauri and a browser-based frontend
* Integrate popular JS frameworks or Rust WASM frameworks with Tauri
* Create UI elements such as windows, menus, and tray icons
* Connect a Tauri app to a database (PostgreSQL or SQL Server)
* Connect a Tauri app to a REST API
* Publish and distribute a Tauri app

**Outline**

* Introduction
* Desktop Apps with Tauri
	+ What is Tauri?
	+ How does it compare to Electron, Qt, and WinForms?
* Getting Started
	+ Create a Tauri Project
	+ Integrate with JavaScript or Rust-based browser UI framework
	+ Run and Debug Tauri Applications
* Call Backend Rust Code from the Front-End
	+ Define Rust Backend Functions
	+ Call Rust Backend Functions
	+ Pass Arguments to Rust Backend Functions
	+ Return Data from Rust Backend Functions
	+ Handle Errors from Rust Backend Functions
	+ Asynchronous Rust Backend Functions
* Interact with App Parts from the Rust Backend
	+ Window
	+ AppHandle
	+ Managed State
	+ Multiple Commands
* UI Elements
	+ Window
	+ Menu
	+ Tray Icon
	+ Splash Screen
	+ Dialog
	+ Notification
* Interacting with the Operating System
	+ Create a File System Dialog
	+ Create a File System Watcher
* Database Programming
	+ What is a Database?
	+ Connect to Postgresql
	+ Query data from the database
	+ Modify data in the database
	+ Connect Tauri to a Database
* Deployment
	+ Create Platform Specific Installers
	+ Cross-Platform Compilation with CI/CD
	+ Signing the Application
* Conclusion