

**Building React Apps with Remix**

**Course Number:** RCT-138
**Duration:** 5 days

**Overview**

This React Apps with Remix training course teaches attendees to how to deploy a fully functional React application using Remix. Participants master Remix's architecture, including React Router and server-side handling using Node.js and Visual Studio Code.

**Prerequisites**

All attendees must have experience with modern JavaScript or TypeScript, including the new language features like classes, modules, arrow functions, and destructuring.

**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 Needed on Each Student PC**

Students will need a free, personal GitHub account to access the courseware and permission to install Node.js and Visual Studio Code on their computers. Also, students will need permission to install NPM Packages and Visual Studio Extensions. A cloud-based environment can be provided if students cannot configure a local environment.

**Objectives**

* Discover why React and Remix are used for modern web development and the specific problems they solve
* Understand Remix’s architecture
* Set up a development environment, including installing Node.js and configuring Visual Studio Code
* Create and manage a Remix project
* Understand folder structures, browser support, handling styles and assets, and managing dependencies
* Create and render React components, understand JSX, and optimize rendering with keys and various JSX operators
* Master advanced concepts like component props, events, and hooks, including state, effect, callback, and custom hooks.
* Explore Remix’s full-stack capabilities
* Implement styling techniques (CSS Modules, Tailwind CSS, CSS-in-JS) and unit testing (Jest, Testing Library)
* Deploy a fully functional React app with Remix

**Outline**

* Introduction
* React and Remix Overview
	+ Why React and Remix?
	+ What problem does React solve?
	+ What problem does Remix solve?
* Remix Architecture
	+ React Router
	+ Compiler
	+ Server-Side HTTP Handler
	+ Web Fetch API
	+ Server Framework
	+ Browser Framework
* Development Environment
	+ Install Node.js
	+ Configure Visual Studio Code
	+ Install React Developer Tools
	+ Install Remix NPM Packages
	+ Remix CLI
* Remix Project Setup
	+ Create a new project
	+ Folder Structure
	+ Browser Support
	+ Styles and Assets
	+ Dependencies
* React Components
	+ Creating an Element
	+ Create a Function Component
	+ Rendering a Component
	+ Composing & Reuse
* React Component Rendering and JSX
	+ What problem does JSX solve?
	+ Embedding Expressions
	+ Specifying Attributes
	+ Using Fragments
	+ Virtual DOM and Fiber Nodes
	+ Ternary Operator (?)
	+ Logical (&& and ||) Operators
	+ Rendering a list of data
	+ Optimizing rendering with keys
* React Component Props
	+ Immutable Props
	+ String Literals vs. Expressions
	+ Prop Types
	+ Default Prop Values
	+ Naming Patterns for Props
* React Component Events
	+ What are Events?
	+ Common Events in React: Click and Change
	+ Event Handlers and Functional Component
	+ Passing Event Handlers via Props
* React Component Hooks
	+ What is Component State?
	+ State Hook
	+ Effect Hook
	+ Callback Hook
	+ Custom Hooks
* Capture Data with Forms
	+ Controlled and Uncontrolled Components
	+ Enable Change Logic across Multiple Form Controls
	+ Wiring up Input, Textarea, and Select
	+ Handling different types of Input
* React Component Architecture
	+ Reusable Components
	+ Component Communication
	+ Design Patterns
	+ Container and Presentation Components
	+ Defining Prop Drilling
* React Router
	+ Define Routes
	+ Pages and Layouts
	+ Linking and Navigating
	+ Dynamic Routes
	+ Error Handling
	+ File Structure
* Remix Full Stack Data Flow
	+ Loaders
	+ Components
	+ Actions
	+ Submission and Revalidation
* Isomorphic Rendering
	+ Server vs. Client Execution
	+ Server Rendering
	+ Client Rendering
	+ Server and Client Composition Patterns
* Progressive Enhancement
	+ What is Progressive Enhancement?
	+ Why Progressive Enhancement?
	+ Progressive Enhancement and Remix
	+ Performance
	+ Resilience and Accessibility
	+ Simplicity
* Pending UI
	+ What is a Pending UI?
	+ Busy Indicators
	+ Optimistic UI
	+ Skeleton Fallbacks
* State Management
	+ What is State Management?
	+ React and State Management
	+ React Anti-Patterns in Remix
	+ Managing State with Remix instead of React
* Network Concurrency Management
* Form vs. Fetcher
	+ Form Component
	+ useActionData Hook
	+ useFetcher Hook
	+ useNavigation Hook
	+ URL Considerations
	+ When the URL Should Change
	+ When the URL Should Not Change
* Styling
	+ CSS Modules
	+ Tailwind CSS
	+ CSS-in-JS (Styled Components)
	+ Sass
* Unit Testing Overview
	+ Jest and Testing Library
	+ What are React components tested for?
	+ Tests, Test Suites, Assertions, and Mocking
	+ Test DOM rendering
	+ Test Event Handlers with Spies
	+ Test Custom Hooks
	+ Mocking Components
	+ Mocking Hooks
* Deployment
* Conclusion