To discuss this course and customizations:
Call: +1 434-509-5680 or Email: sales@training4programmers.com

React Apps with Remix

Duration

5 Days

Description

This comprehensive course on “React Apps with Remix” offers an in-depth exploration of modern web development using React and Remix, highlighting their unique solutions to contemporary web development challenges. Participants will delve into the intricate architecture of Remix, including key aspects like React Router and server-side handling, while mastering the setup of a robust development environment with tools like Node.js and Visual Studio Code. The course emphasizes practical skills in creating and managing Remix projects, from understanding folder structures to managing styles and dependencies. Students will also gain hands-on experience in advanced React concepts, such as component props, events, and hooks, and explore full-stack development features like isomorphic rendering and progressive enhancement. The curriculum culminates in teaching essential styling techniques, effective unit testing, and deploying a fully functional React application using Remix.

Objectives

- Grasp why React and Remix are chosen for modern web development and the specific problems they solve.
- Dive into Remix’s architecture, including React Router, Compiler, server-side HTTP handling, Web Fetch API, and understanding both server and browser frameworks.
- Master setting up a development environment, including installing Node.js, configuring Visual Studio Code, React Developer Tools, and Remix NPM packages.
- Learn to create and manage a Remix project, understanding folder structures, browser support, handling styles and assets, and managing dependencies.
- Gain proficiency in creating and rendering React components, understanding JSX, and optimizing 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 including isomorphic rendering, progressive enhancement, pending UI concepts, state management, and network concurrency.
- Learn styling techniques (CSS Modules, Tailwind CSS, CSS-in-JS), unit testing (Jest, Testing Library), and deploy a fully functional React app with Remix.

Training 4 Programmers

To discuss this course and customizations:
Call: +1 434-509-5680 or Email: sales@training4programmers.com

Prerequisites

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

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 Node.js and Visual Studio Code on their computers. Also, students will need permission to install NPM Packages and Visual Studio Extensions. If students are unable to configure a local environment, a cloud-based environment can be provided.

Outline

- Introduction
- React and Remix Overview
  - Why React and Remix?
  - What problem does React solve?
  - What problem does Remix solve?
- Remix Architecture
  - React Router
  - Compuler
  - 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
  - Resillence 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