

**Comprehensive React and Redux using TypeScript**

**Course Number:** RCT-120
**Duration:** 5 days

**Overview**

[React](https://reactjs.org/) is a JavaScript library for building web user interfaces and is one of the most wanted and loved web frameworks today. React makes it painless to create interactive UIs on top of web standards and is often used with the Redux library for managing the data or state in a React application.

In this React course, students gain hands-on experience with the latest version of React and the tools for developing React applications. Developers learn the skills they need to immediately build React and Redux applications using the TypeScript language.

**Note:** This React class can also be taught using JavaScript/ECMAScript.

**Prerequisites**

Students must have object-oriented programming experience. Some experience with JavaScript is helpful; the new language features of JavaScript and TypeScript are covered/reviewed in class.

**Materials**

All attendees receive comprehensive courseware.

**Software Needed on Each Student PC**

* Google Chrome and/or Firefox
* Other modern browsers as desired
* IDE/development environment of your choice
* Other free software and lab files that Accelebrate would specify, including Node.js

**Objectives**

* Use a JavaScript package manager (either npm or Yarn)
* Quickly understand the new JavaScript language features, including classes, modules, and arrow functions
* Articulate what React is and why it is useful
* Explore the basic architecture of a React application
* Gain a deep understanding of JSX and the Virtual DOM
* Use React components to build interactive interfaces
* Create and validate forms using controlled components
* Make HTTP calls to read or change data
* Explore the common component architecture patterns
* Configure simple and complex routing
* Understand state management including when it is needed and the various alternatives
* Utilize Redux to manage the state of the application
* Use React and Redux together
* Implement React and Redux best practices
* Write tests for React using Jest and React Testing Library

**Outline**

* Introduction
* Package Manager (npm/Yarn)
	+ npm
		- What is Node.js?
		- What is npm?
		- Using npm
		- npm Scripts
	+ Yarn
		- Why Yarn?
		- npm vs. Yarn
		- Installation
		- Using Yarn
* Compiler Setup
	+ Setup TypeScript Compiler (tsc)
* ES6/ES2015
	+ Classes
	+ Scope (var, let, const)
	+ Arrow Functions
	+ Modules
	+ Template Literals
	+ Default, Rest, Spread
	+ Default
	+ Rest
	+ Spread
	+ Destructuring
	+ Optional Parameters
	+ Object.assign()
	+ Object Initializer
* TypeScript
	+ Understanding TypeScript, ES6, and ES5
	+ How TypeScript Works
	+ Who is Behind TypeScript?
	+ Type Annotations
	+ Classes
		- Fields
		- Parameter Properties
		- Member visibility (public, private, protected)
	+ Automatically Import Modules
	+ Nullable Parameters
	+ Interfaces
	+ Generics
* Project Setup (Create React App)
	+ Create new Project
	+ Folder Structure
	+ Browser Support
	+ Styles and Assets
	+ Dependencies
* Best Practices (Code Organization & Conventions)
* React Overview
	+ Why React?
	+ What it is?
	+ Why it is useful?
	+ Angular, React Compared
	+ Web application architectures
		- Server-side web application architecture
		- Single-page web application architecture
	+ React Architecture
* Elements
	+ Hello World in JavaScript
	+ Hello World in React
* JSX
	+ Replacing createElement
	+ Embedding Expressions
	+ Specifying Attributes
* Virtual DOM
* Components
	+ Creating an Element
	+ Create a Function Component
	+ Rendering a Component
	+ Creating a Class Component
	+ Composing & Reuse
* Props
	+ Read-only
	+ String Literals vs. Expressions
	+ Function vs. Class Components
* Events
	+ Listening/Subscribing/Wiring to an Event
	+ In Vanilla JavaScript
	+ In React: Function Component
	+ In React: Class Component
	+ Binding
		- Why Binding is Necessary?
		- Class Method
		- Arrow Function
	+ Passing Parameters
		- Using Arrow Functions
		- Using Bind
	+ Handling Events
		- Using Arrow Functions
		- Using Bind
		- Synthetic Events
* Hooks
	+ Definition
	+ Why Hooks?
	+ No Breaking Changes
	+ Hooks API: useState, useEffect, useRef, useContext
	+ Rules of Hooks
* State
	+ Definition
	+ State in Function Components
	+ useState Hook
	+ State in Class Components
	+ Mutating state with setState
	+ Using State Correctly
	+ Data Flows Down
* Side Effects & Lifecycle
	+ useEffect Hook
	+ What does useEffect do?
	+ Using useEffect
	+ useEffect Demo
	+ What are Lifecycle Methods
	+ Understanding Mounting
	+ Common vs. Less Common Methods
	+ Using Lifecycle Methods
* Conditional Rendering
	+ If, else
	+ Conditional Operator (?)
	+ Logical (&&) Operator
* Lists
	+ In Vanilla JavaScript: for loop, array.forEach, array.map
	+ In React: using Elements, Components
	+ Why Keys are Needed
* Component Architecture
	+ Reuse
	+ Component Communication
	+ Design Patterns
		- Container and Presentation Components
		- Composition vs. Inheritance
* Forms
	+ Controlled Components
	+ Reuse of Change Logic across Multiple Inputs
	+ Handling Form Submission
	+ Controlling Other Form Elements: select, textarea, number
	+ Validation
	+ Uncontrolled Components
* HTTP
	+ Axios library
	+ Fetch API
	+ Using with React (HTTP GET)
	+ Refactoring for Reuse
	+ HTTP POST, PUT, DELETE
* Routing (React Router)
	+ Installation
	+ Basics
	+ Handling Not Found (404)
	+ Parameters (Url & Query)
	+ Nesting
* Hooks
	+ Defined
	+ Why Hooks?
	+ No Breaking API Changes
	+ Hooks API
	+ useState
	+ useEffect
	+ Custom Hooks
	+ Rules of Hooks
* Build & Deploy
	+ Building a React Application for Production
	+ Deploying a React Application
	+ Serving Apps with Client-Side Routing
	+ Customizing Environment Variables
* Custom Hooks
	+ Background
	+ Definition
	+ Reuse of stateful logic
* Context
	+ When to use Context
	+ useContext Hook
* State Management
	+ What is State?
	+ When do you need Redux?
	+ Alternatives to Redux
	+ Using Context for Shared State
	+ Server State: React Query, SWR, or GraphQL client
* Redux
	+ What is Redux?
	+ What is State?
	+ Benefits Checklist
	+ Principles of Redux
	+ Core Concepts (Store, State, Reducers, Actions, Action Creators)
	+ Complementary Packages
	+ When do you need Redux?
	+ Basic Redux Example (includes time traveling)
	+ Gotchas/Tips
* Using Redux with React (React Redux Library)
	+ Redux with React in Function Components
	+ useSelector and useDispatch Hooks
	+ Provider
	+ Example
	+ Redux with React in Class Components
	+ Higher-Order Components
	+ The connect function
	+ Writing mapState functions
	+ Writing mapDispatch Functions
	+ Example
* Asynchronous Actions (Redux Thunk)
	+ Overview
	+ Async Actions (Thunks)
	+ Installation
	+ Your First Thunk
	+ Full CRUD Example
* Putting It All Together (React & Redux & Thunk)
* Testing
	+ Tools (Jest, React Testing Library or Enzyme, JSDOM)
	+ Syntax
	+ Testing Vanilla JavaScript with Jest
	+ Mocking
		- Mocking Modules
		- Mocking Functions
	+ Debugging Tests
	+ Component Tests
		- Shallow
		- Full
		- Snapshot
	+ Testing Redux Actions & Thunks, Reducers, HTTP
* Optional Topics/FAQ (if time permits)
	+ Styling
		- Using Classes
		- Using Styles
		- CSS in JS Overview
	+ Performance
		- What causes a component to render?
		- Wasted Renders
		- Memoization
		- React.memo
		- Pure Components
	+ Advanced Components
		- Higher-Order Components
		- Render Props
* Conclusion