Cloud Contraptions
TRAINING & CONSULTING

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

React Apps with TanStack Start

Class Duration

21 hours of live training delivered over 3 days

Student Prerequisites

- Solid working experience building React applications with function components and hooks
- Experience with TypeScript and modern JavaScript (modules, async/await, destructuring)
- Prior experience with another React framework (Next.js, Remix) helpful but not required

Target Audience

React developers and teams evaluating or adopting TanStack Start as their full-stack framework. Ideal for teams that want end-to-end TypeScript type safety, a client-first architecture, and freedom to deploy anywhere — and for teams comparing TanStack Start against Next.js before committing. Participants should be comfortable building React applications; this course teaches the framework, not React fundamentals.

Description

This course teaches **TanStack Start**, the full-stack React framework built on TanStack Router. Start has reached the release-candidate stage — feature-complete with a stable API, with the 1.0 release expected imminently — and is positioned as the type-safe, client-first alternative to Next.js: instead of organizing your application around React Server Components, Start gives you a fully type-safe router, type-safe **server functions** (RPCs between client and server), full-document **SSR with streaming**, and standard Vite or Rsbuild builds that deploy to any modern host. Participants master TanStack Router's file-based routing — typed path params, typed search params, nested layouts, loaders, and prefetching — then add the server pieces: server functions with middleware, server routes for API endpoints, selective SSR, SPA mode, static prerendering, and incremental static regeneration. Data loading is taught the way production Start apps are built: route loaders integrated with **TanStack Query** for caching, mutations, and optimistic

updates. The course finishes with forms backed by server functions, error and not-found boundaries, authentication patterns, and deployment. Because the framework is young, the course is explicit about maturity: what is stable, what is experimental (React Server Components support), and how to track the project as it evolves.

Learning Outcomes

- Explain where TanStack Start fits in 2026: its release-candidate maturity, its client-first architecture, and how it compares to Next.js.
- Scaffold and structure a TanStack Start project with Vite, file-based routing, and TypeScript end-to-end.
- Master TanStack Router: typed path params, validated search params, nested layouts, navigation, and prefetching.
- Load data with route loaders and integrate TanStack Query for caching, mutations, and optimistic updates.
- Write type-safe server functions for queries and mutations, and compose them with middleware for auth, logging, and context.
- Build API endpoints with server routes alongside the application.
- Control rendering per route: full-document SSR, streaming, selective SSR, SPA mode, static prerendering, and ISR.
- Build forms on top of server functions with validation and progressive-enhancement-friendly patterns.
- Handle errors with route-level error and not-found boundaries.
- Deploy Start applications to Cloudflare, Netlify, Vercel, or self-hosted Node, and test with Vitest and Playwright.

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 Code Extensions. If students are unable to configure a local environment, a cloud-based environment can be provided.

Cloud Contraptions
TRAINING & CONSULTING

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

Training Topics

Introduction

- What is TanStack Start, and what problem does it solve?
- Client-first architecture vs. the React Server Components model
- Start vs. Next.js: type safety, deployment freedom, mental model
- Maturity in 2026: release candidate, stable API, the road to 1.0
- What is experimental (React Server Components support) and what is not

Project Setup

- Scaffolding a Start project
- Vite (and Rsbuild) as the build foundation
- Project structure: routes, server, client entry points
- TypeScript configuration and the generated route tree
- Environment variables and client/server import protection

TanStack Router: Routing

- File-based routing conventions
- Typed path parameters
- Search params as first-class, validated, typed state
- Nested routes and layouts
- Route context
- Navigation with `Link` and `useNavigate`
- Preloading and prefetching strategies

Data Loading

- Route loaders: load before render, no waterfalls
- Loader caching, staleness, and invalidation
- Deferred data and streaming from loaders
- Integrating TanStack Query: `queryOptions`, `ensureQueryData` in loaders
- Mutations and optimistic updates with TanStack Query
- When to use loaders alone vs. loaders + Query

Server Functions

- Type-safe RPCs with `createServerFn`
- GET vs. POST server functions: data fetching and mutations in one API
- Input validation (Zod) with inferred types end-to-end

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

- Calling server functions from loaders, components, and hooks
- Static (build-time) server functions

Middleware and Server Routes

- Middleware for server functions: auth, logging, request context
- Composing and chaining middleware
- Server routes: building API endpoints alongside the app
- Cookies, headers, sessions, and authentication patterns

Rendering Strategies

- Full-document SSR and hydration
- Streaming responses
- Selective SSR: opting routes out
- SPA mode for fully client-rendered apps
- Static prerendering
- Incremental static regeneration (ISR)

Forms

- Forms backed by server functions
- Validation on client and server with shared schemas
- Pending, error, and optimistic UI states
- TanStack Form overview: typed, framework-agnostic form state

Errors and Boundaries

- Route-level error boundaries
- Not-found handling
- Hydration errors: diagnosing and fixing

Testing

- Unit and component testing with Vitest + Testing Library
- Testing server functions
- End-to-end testing with Playwright

Deployment

- Build outputs for Node, Cloudflare, Netlify, and Vercel
- Hosting configuration and runtime targets
- CDN asset URLs and caching