Cloud Contraptions
TRAINING & CONSULTING

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

# Advanced React

## 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)
- Familiarity with `useState`, `useEffect`, props, and component composition is assumed

## Target Audience

Experienced React developers who can ship features but want to master what changed in React 19 and why. Ideal for teams maintaining large client-side React codebases who need to adopt the React Compiler, untangle state management, eliminate performance problems, and modernize patterns written for React 16–18. This is a client-side deep dive: teams building on Next.js and React Server Components should also see our **React Apps with Next.js** course, which covers the server side of React 19.

## Description

This advanced course takes experienced React developers deep into React 19 and the modern client-side React platform. Participants master the React 19 feature set — **Actions**, the **`use`** hook, **`useActionState`**, **`useOptimistic`**, and **`useTransition`** — and learn how the **React Compiler** (stable since late 2025) replaces the manual `useMemo` / `useCallback` / `memo` memoization discipline that defined React performance work for years. The course covers Suspense patterns and concurrent rendering in depth, then turns to disciplined performance engineering: profiling with the React DevTools Profiler, finding real render bottlenecks, and fixing them with evidence rather than folklore. Participants learn a clear state architecture — server state with **TanStack Query**, client state with **Zustand** and context — along with custom hooks design, error boundary strategies, and testing advanced asynchronous patterns with **Vitest** and React Testing Library. React Server Components

and server-side rendering are referenced where they shape client-side decisions, but the server-side stack is covered in our companion Next.js course rather than duplicated here.

## Learning Outcomes

- Apply the React 19 feature set: Actions for mutations, `useActionState` and `useFormStatus` for form state, `useOptimistic` for optimistic UI, and the `use` hook for promises and context.
- Adopt the React Compiler and retire manual `useMemo` / `useCallback` / `memo` discipline — and recognize the cases where manual memoization still matters.
- Use `useTransition` and `useDeferredValue` to keep interfaces responsive under concurrent rendering.
- Design Suspense boundaries for data loading, code splitting, and coordinated loading states.
- Profile applications with the React DevTools Profiler, interpret flame graphs and commit timelines, and fix measured bottlenecks.
- Architect state deliberately: TanStack Query for server state (caching, invalidation, optimistic updates) and Zustand or context for genuine client state.
- Design custom hooks with clean contracts, stable identities, and TypeScript generics.
- Build layered error boundary strategies that recover gracefully, including errors thrown from Actions and Suspense.
- Test advanced patterns — transitions, Suspense, optimistic updates, custom hooks — with Vitest and React Testing Library.

## 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

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

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

## Training Topics

### Introduction

- React 19 in 2026: what changed and why it matters
- The 19.x release line: Actions, the Compiler, and concurrent features
- Where Server Components fit (and where this course hands off to the Next.js course)

### React 19 Actions

- Actions: async transitions for data mutations
- `useActionState` for form state, pending status, and results
- `useFormStatus` in design-system components
- `useOptimistic` for instant, self-correcting UI
- Error handling and rollback in Actions

### The `use` Hook

- Reading promises with `use` and Suspense
- Reading context conditionally with `use`
- Promise caching: why you don’t create promises in render
- Patterns and pitfalls compared to `useEffect`-based fetching

### The React Compiler

- What the Compiler does: automatic, fine-grained memoization
- The manual discipline it replaces: `useMemo`, `useCallback`, `React.memo`
- The Rules of React and `eslint-plugin-react-hooks`
- Adopting the Compiler in an existing codebase
- When manual memoization still earns its keep
- Verifying Compiler output in React DevTools

### Concurrent Rendering

- How concurrent rendering works: interruptible renders and lanes
- `useTransition` for non-blocking updates
- `useDeferredValue` for expensive derived UI
- Keeping inputs responsive during heavy renders

Cloud Contraptions
TRAINING & CONSULTING

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

## Suspense Patterns

- Suspense for data loading and code splitting (`lazy`)
- Designing boundary placement: granular vs. coordinated fallbacks
- Avoiding waterfalls: parallel data loading
- Suspense with transitions: avoiding fallback flicker

## Performance Profiling

- React DevTools Profiler: flame graphs, ranked charts, commit timelines
- Finding unnecessary re-renders with evidence
- Component-level vs. structural fixes (composition, children, state colocation)
- Measuring with the browser Performance panel and Web Vitals
- Virtualizing long lists

## State Architecture

- Server state vs. client state: the most important distinction
- TanStack Query: queries, mutations, cache invalidation, optimistic updates
- Zustand for client state: stores, selectors, slices
- Context: what it is good for, and the re-render costs
- `useReducer` and state machines for complex local state
- Choosing among Query, Zustand, context, and component state

## Custom Hooks Design

- Designing hook contracts and return shapes
- TypeScript generics in reusable hooks
- Stable function identities and the Compiler
- Composing hooks; avoiding hidden coupling
- `useSyncExternalStore` for external sources

## Error Boundaries

- Error boundary placement strategy: layered recovery
- `react-error-boundary`: reset, retry, and fallback UX
- Errors from Actions, event handlers, and async code
- Reporting errors with `onCaughtError` / `onUncaughtError`

## Testing Advanced Patterns

- Vitest + React Testing Library setup for React 19


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

- Testing transitions, Suspense fallbacks, and optimistic updates
- Testing custom hooks with `renderHook`
- Mocking TanStack Query and network boundaries (MSW)
- Testing error boundaries and recovery flows