www.logicalimagination.com
800.657.1494

React Fundamentals

Course #: RA-510 **Duration:** 5 days

Prerequisites

Students should have a good understanding of HTML and CSS and be experienced JavaScript developers, with an advanced understanding of JavaScript objects and functions as first class citizens.

Details

React is a JavaScript library for building interactive user interfaces. It can be incrementally adopted, scaling easily from being used as a library to add small functionality to web pages to being used as a framework for a complete single-page-application. React also provides hooks that allow easy interaction with other JavaScript libraries and frameworks.

This course will give the student a solid and in-depth foundation for building applications that use the powerful features provided by React.

Software Needed

- Node.js installed
- Any text editor (we recommend Visual Studio Code: https://code.visualstudio.com/)
- At least one web browser (Google Chrome is recommended)
- Internet connection

Outline

- **Introduction**
  - What is React
  - Why use React?
  - SPAs and React Web Apps
  - NextGen JavaScript Features
- **React Syntax and Basics**
  - Using Create React App
  - Component Basics
  - JSX
  - Props and Dynamic Content
  - State and Event Handling
  - Two-Way Binding
- **Dynamic Content**
  - Conditionally Rendering Content
  - Collection Content
  - Updating State Immutably
  - Collections and Keys
  - More Flexible Collections

- **Styling Content**
  - Inline Styles
  - Dynamically Setting Styles
  - Dynamically Setting Class Names
  - Using Radium
  - Using CSS Modules
- **Debugging**
  - Understanding Error Messages
  - DevTools and Sourcemaps
  - React Developer Tools
  - Error Boundaries
- **Components**
  - Creating Components
  - Stateless vs. Stateful
  - Component Lifecycle
  - Pure Components
  - Higher-Order Components
  - Validating Props
  - Context API
- **Web Server Interactions**
  - AJAX Calls
  - Using Axios
  - Rendering Fetched Data
  - Avoiding Infinite Loops
  - POSTing Data
  - Handling Errors Locally
  - Interceptors
- **Routing**
  - Setting Up the Router Package
  - Rendering Components for Routes
  - Using Routing-Related Props
  - Absolute vs. Relative Paths
  - Nested Routes
  - Route Guards
  - Routing and Deployment
- **Forms**
  - Custom Dynamic Input Components
  - Configuring a Form
  - Handling Form Submission
  - Custom Validation
  - Showing Error Messages
- **Managing State with Redux**
  - Complexity of State Management
  - How Redux Works
  - Reducer Functions and State Store
  - Dispatching Actions
  - Creating Subscriptions
  - Connecting React to Redux
  - Dispatching Actions from Components
- **Async Redux**
  - Adding Middleware
  - Redux Devtools
  - Action Creators
  - Handling Async Actions
  - Action Creators and Get State
- **Testing**
  - Required Testing Tools
  - What to Test?
  - Testing Components
  - Jest and Enzyme

- Testing Containers
- Testing Redux

- **Transitions and Animations**
  - Using CSS Transitions
  - Using CSS Animations
  - ReactTransitionGroup
  - Using the Transition Component
  - Wrapping the Transition Component
  - Animation Timing
  - Transition Events

- **Introduction to Hooks**
  - What are React Hooks?
  - Getting Started with useState()
  - Updating State
  - Multiple States
  - Rules of Hooks
  - Passing State Across Components

- **Side Effects**
  - Sending HTTP Requests
  - useEffect() and Loading Data
  - Understanding useEffect() Dependencies
  - What is useCallback()?
  - Refs and useRef()
  - Cleaning up with useEffect()

- **State Batching**
  - Understanding useReducer()
  - useReducer() and HTTP State
  - Working with useContext()
  - Performance Optimization with useMemo()

- **Custom Hooks**
  - Getting Started
  - Sharing Data with Components
  - Using a Custom Hook