Over 35 Years Of Technology Training

Document Generated: 07/27/2024

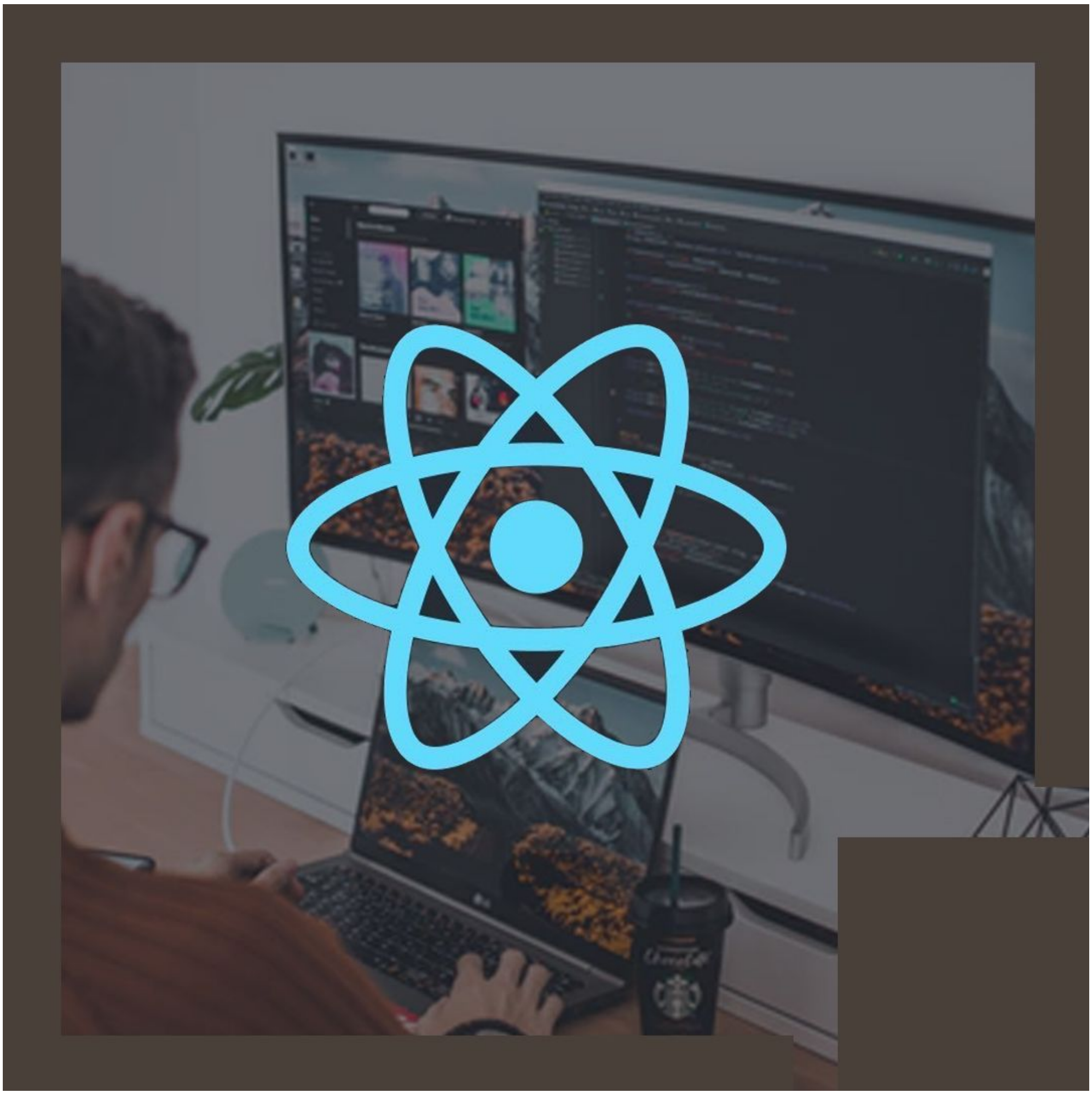
Learning Style: Virtual Classroom

Provider:

Difficulty: Beginner

Course Duration: 5 Days

Mastering React | React, Redux, JSX, Flux, Forms, Unit Testing & More (TTSREACT3)



About this course:

Mastering React is a Five-days, in-depth hands-on training program aimed at becoming the single most valuable tool for getting started with Respond quickly. This training is built for more professional web Developers and gives learners the key skills and hands-on experience they need to create secure, efficient React apps.

You will have a good knowledge of the basics of React after the first few modules, and will also be able to create a large variety of beautiful, immersive web applications with the framework. Topics such as client-side routing between sites, complex state management, and strong on-scale application program interface interaction are also addressed. In a progressive, example-driven method we cover all of the fundamentals. You can build your first software, discover how components can be written, and start managing user interaction. We will also discuss the internal workings of Build React App (Facebook's software to run React applications), write automated unit testing, and develop a multi-page application that utilizes routing on the client-side.

The above part of this training is moving into more advanced principles that you can see used in broad apps for growth. Both principles address data infrastructure, storage, and management strategies: Redux is a state management model based on the Flux infrastructure from Facebook. Redux offers a framework for huge state trees and lets you decouple client interaction from state changes in your device.

The React Js Developer can earn an average salary of $112,250 per annum.

Course Objective:

Working within a focused, hands-on training framework, learners should able to:

- How to create and Install first React component
- Using React to handle User Interface elements, react to user feedback and build a state of the art components
- Comprehend the FLUX structure and use FLUX with react to construct an app
- Discover methods to test the Code for ReactJS
- Understand the fundamentals of ReactJS by taking an overview
- Using some main lifecycle events to incorporate and learn about ES6 syntaxes in the reaction environment
- Consider a system reusable with validation aids and appropriately organize the components
- Use JSX to Refactor the ReactJS component
- Use webpack to deploy your code

Audience:

- It is an introductory-level online training program for web.dev who want to further improve their expertise in modern web development. To excel in this class

Prerequisite:

- Participants are expected to have up-to-date, hands-on, solid web app development experience, and to be versed in critical JavaScript, CSS3, and HTML5.

Course Outline:

Module 1: Your first React Web Application

- Building Product Hunt
- Setting up your development environment
    - Code editor
    - Node.js and npm
    - Install Git
    - Browser
- Special instruction for Windows users
    - Ensure IIS is installed
- JavaScript ES6/ES7
- Getting started
    - Sample Code
    - Previewing the application
    - Prepare the app
- What’s a component?
    - Our first component
    - JSX
    - The developer console
    - Babel
    - ReactDOM.render()
- Building Product
- Making Product data-driven
    - The data model
    - Using props
    - Rendering multiple products
- React the vote (your app’s first interaction)
    - Propagating the event
    - Binding custom component methods
    - Using state
    - Setting state with this.setState()
- Updating state and immutability
- Refactoring with the Babel plugin transform-class-properties
    - Babel plugins and presets
    - Property initializers
    - Refactoring Product
    - Refactoring ProductList

Module 2: Components

- A time-logging app
- 1. Getting started
    - Previewing the app
    - Prepare the app
- Breaking the app into components
- The steps for building React apps from scratch
- Step 2: Build a static version of the app
    - TimersDashboard
    - EditableTimer
    - TimerForm
    - ToggleableTimerForm
    - Timer
    - Render the app
- Step 3: Determine what should be stateful
    - State criteria
    - Applying the criteria
- Step 4: Determine in which component each piece of state should live
    - The list of timers and properties of each timer
    - Whether or not the edit form of a timer is open
    - Visibility of the create form
- Step 5: Hard-code initial states
    - Adding state to TimersDashboard
    - Receiving props in EditableTimerList
    - Props vs. state
    - Adding state to EditableTimer
    - Timer remains stateless
    - Adding state to ToggleableTimerForm
    - Adding state to TimerForm
- Step 6: Add inverse data flow
    - TimerForm
    - ToggleableTimerForm
    - TimersDashboard
- Updating timers
    - Adding editability to Timer
    - Updating EditableTimer
    - Updating EditableTimerList
    - Defining onEditFormSubmit() in TimersDashboard
- Deleting timers
    - Adding the event handler to Timer
    - Routing through EditableTimer
    - Routing through EditableTimerList
    - Implementing the delete function in TimersDashboard
- Adding timing functionality.
    - Adding a forceUpdate() interval to Timer
- Add start and stop functionality
    - Add timer action events to Timer
    - Create TimerActionButton
    - Run the events through EditableTimer and EditableTimerList
- Methodology review

Module 3: JSX and the Virtual DOM

- React Uses a Virtual DOM
- Why Not Modify the Actual DOM?
- What is a Virtual DOM?
- Virtual DOM Pieces
- ReactElement
  - Experimenting with ReactElement
  - Rendering Our ReactElement
  - Adding Text (with children)
  - ReactDOM.render()
- JSX
  - JSX Creates Elements
  - JSX Attribute Expressions
  - JSX Conditional Child Expressions
  - JSX Boolean Attributes
  - JSX Comments
  - JSX Spread Syntax
  - JSX Gotchas
  - JSX Summary

Module 4: Advanced Component Configuration with props, state, and children

- Intro
- How to use this chapter
- ReactComponent
  - Creating ReactComponents - createReactClass or ES6 Classes
  - render() Returns a ReactElement Tree
  - Getting Data into render()
- props are the parameters
- PropTypes
- Default props with getDefaultProps()
- context
- state
  - Using state: Building a Custom Radio Button
  - Stateful components
  - State updates that depend on the current state
  - Thinking About State
- Stateless Components
  - Switching to Stateless
  - Stateless Encourages Reuse
- Talking to Children Components with props.children
  - React.Children.map() & React.Children.forEach()
  - React.Children.toArray()

Module 5: Forms

- Forms 101
  - Preparation
  - The Basic Button.

    - Events and Event Handlers
    - Back to the Button
- Text Input
    - Accessing User Input With refs .
    - Using User Input.
    - Uncontrolled vs. Controlled Components
    - Accessing User Input With state
    - Multiple Fields
    - On Validation
    - Adding Validation to Our App.
    - Creating the Field Component
    - Using our new Field Component
- Remote Data.
    - Building the Custom Component
    - Adding CourseSelect .
    - Separation of View and State.
- Async Persistence .
- Redux .
- Form Component.
    - Connect the Store.
    - Form Modules
    - formsy-react .
- react-input-enhancements
- tcomb-form
- winterfell
- react-redux-form.

Module 6: Unit Testing.

- Writing tests without a framework .
    - Preparing Modash.
    - Writing the first spec .
    - The assertEqual() function.
- What is Jest?.
- Using Jest.
    - expect().
    - The first Jest test for Modash.
    - The other truncate() spec
    - The rest of the specs
- Testing strategies for React applications.
    - Integration vs Unit Testing.
    - Shallow rendering
    - Enzyme
- Testing a basic React component with Enzyme
    - Setup.
    - The App component
    - The first spec for App .
    - More assertions for App
    - Using beforeEach
    - Simulating a change .

- Clearing the input field
- Simulating a form submission
- Writing tests for the food lookup app
    - FoodSearch
    - Exploring FoodSearch
- Writing FoodSearch.test.js
    - In initial state
    - A user has typed a value into the search field
    - Mocking with Jest
    - Mocking Client.
    - The API returns results
    - The user clicks on a food item
    - The API returns empty result set

Module 7: Routing

- What's in a URL?
    - React Router's core components .
- Building the components of react-router.
    - The completed app
    - Building Route
    - Building Link .
    - Building Router
    - Building Redirect
    - Using react-router
    - More Route
    - Using Switch .
- Dynamic routing with React Router
    - The completed app
    - The server's API.
    - Starting point of the app.
    - Using URL params
    - Propagating pathnames as props
    - Dynamic menu items with NavLink.
- Supporting authenticated routes
    - The Client library
    - Implementing login
    - PrivateRoute, a higher-order component.
    - Redirect state .

Module 8: Intro to Flux and Redux

- Why Flux?
    - Flux is a Design Pattern.
    - Flux overview .
- Flux implementations
- Redux .
- Building a counter
    - Preparation
    - Overview.

    - The counter’s actions .
    - Incrementing the counter
    - Decrementing the counter
    - Supporting additional parameters on actions
- Building the store
- The core of Redux
- The beginnings of a chat app
    - Previewing
    - State
    - Actions
- Building the reducer() .
    - Initializing state.
    - Handling the ADD_MESSAGE action
    - Handling the DELETE_MESSAGE action
- Subscribing to the store .
    - createStore() in full
- Connecting Redux to React.
    - Using store.getState().
    - Using store.subscribe()
    - Using store.dispatch().
    - The app’s components
    - Preparing App.js.
    - The App component
    - The MessageInput component
    - The MessageView component

Module 9: Intermediate Redux .

- Preparation
- Using createStore() from the redux library .
- Representing messages as objects in state.
    - Updating ADD_MESSAGE
    - Updating DELETE_MESSAGE
    - Updating the React components .
- Introducing threads.
    - Supporting threads in initialState
    - Supporting threads in the React components
    - Modifying App
    - Turning MessageView into Thread
- Adding the ThreadTabs component .
    - Updating App .
    - Creating ThreadTabs .
- Supporting threads in the reducer
    - Updating ADD_MESSAGE in the reducer
    - Updating the MessageInput component.
    - Updating DELETE_MESSAGE in the reducer
- Adding the action OPEN_THREAD.
    - The action object.
    - Modifying the reducer
    - Dispatching from ThreadTabs

- Breaking up the reducer function.
    - A new reducer()
    - Updating threadsReducer().
- Adding messagesReducer().
    - Modifying the ADD_MESSAGE action handler .
    - Creating messagesReducer()
    - Modifying the DELETE_MESSAGE action handler
    - Adding DELETE_MESSAGE to messagesReducer().
- Defining the initial state in the reducers
    - Initial state in reducer().
    - Adding initial state to activeThreadIdReducer().
    - Adding initial state to threadsReducer()
- Using combineReducers() from redux.

Module 10: Using Presentational and Container Components with Redux

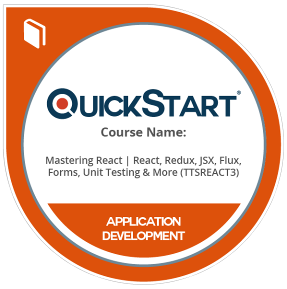
- Presentational and container components.
- Splitting up ThreadTabs
- Splitting up Thread
- Removing store from App
- Generating containers with react-redux
    - The Provider component.
    - Wrapping App in Provider.
    - Using connect() to generate ThreadTabs
    - Using connect() to generate ThreadDisplay
- Action creators

Module 11: (OPTIONAL) Working with React Native.

- Init
- Routing
- <Navigator />
    - renderScene()
    - configureScene()
- Web components vs. Native components
    - <View />.
    - <Text />.
    - <Image />.
    - <TextInput />
    - <TouchableHighlight />, <TouchableOpacity />, and <TouchableWithoutFeedback/>
    - <ActivityIndicator />.
    - <WebView /> .
    - <ScrollView />.
    - <ListView /> .
- Styles .
    - StyleSheet.
    - Flexbox
- HTTP requests
- What is a promise

    - Enter Promises
- Single-use guarantee.
- Creating a promise
- Debugging with React Native

Credly Badge:



Display your Completion Badge And Get The Recognition You Deserve.

Add a completion and readiness badge to your Linkedin profile, Facebook page, or Twitter account to validate your professional and technical expertise. With badges issued and validated by Credly, you can:

- Let anyone verify your completion and achievement by clicking on the badge
- Display your hard work and validate your expertise
- Display each badge's details about specific skills you developed.

Badges are issued by QuickStart and verified through Credly.

[Find Out More] or [See List Of Badges]