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

Modality: Virtual Classroom

Duration: 5 Days

SATV Value:

CLC:

NATU:

SUBSCRIPTION: No

About this course:

Mastering React is a 5-day, in-depth hands-on course that aims to be the single most useful resource on getting up to speed quickly with React. Geared for more experienced web developers, this course provides students with the core knowledge and hands-on skills they require to build reliable, powerful React apps.

After the first few modules, you'll have a solid understanding of React's fundamentals and will be able to build a wide array of rich, interactive web apps with the framework. Topics like client-side routing between pages, managing complex state, and heavy API interaction at scale are also discussed. We cover all the fundamentals with a progressive, example-driven approach. You'll create your first apps, learn how to write components, and start handling user interaction. We will also explore the inner workings of Create React App (Facebook's tool for running React apps), writing automated unit tests, and building a multi-page app that uses client-side routing.

The latter part of this course moves into more advanced concepts that you'll see used in large, production applications. These concepts explore strategies for data architecture, transport, and management: Redux is a state management paradigm based on Facebook's Flux architecture. Redux provides a structure for large state trees and allows you to decouple user interaction in your app from state changes.

The average salary of a React Js Developer salary is **$112,250** per year.

Course Objective:

Working within in an engaging, hands-on learning environment, attendees will learn to:

- Understand the ReactJS basics through an overview
- Install and create your first React component
- Refactor the ReactJS component using JSX
- Handle UI elements events with React, respond to users input, and create stateful components
- Use some core lifecycle events for integration and find out about ES6 syntaxes in the React world

- Understand the FLUX architecture and create an application using FLUX with React
- Make a component more reusable with validation helpers and structure your components properly
- Explore techniques to test your ReactJS code
- Deploy your code using webpack.

Course Topics: This is a high-level list of the course topics covered in this training. Please see the detailed Course Agenda with session details, lessons and labs listed below.

- Your first React Web Application
- Components
- JSX and the Virtual DOM
- Advanced Component Configuration with props, state, and children
- Forms
- Unit Testing.
- Routing
- Intro to Flux and Redux
- Intermediate Redux
- Using Presentational and Container Components with Redux
- (OPTIONAL) Working with React Native.

Audience:

- This is an **introductory-level** React development course, designed for experienced web developers that need to further extend their skills in modern web development. In order to be successful in this class

Prerequisite:

- Attendees are required to have current, hands-on, solid experience in web application development, and be versed in HTML5, CSS3 and JavaScript essentials.

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