MERN STACK Development

By Dr. Vishwanath Rao

Prerequisites

To succeed in this training, participants should have experience with:

- JavaScript programming (ES6 or higher) at an intermediate level including some exposure to exception handling, promises, and debugging Web applications in the browser environment.
- Core front end technologies such as HTTP, HTML, CSS, DOM and browser development tools

Conceptual knowledge of data base systems and web application design is assumed.

Audience

Developers, Technical Leads, and Software Architects.

Duration : 5 Days

MERN Training

1 - React Overview

- What is React?
- What's in a Name?
- React Component Model
- What React Is Not
- What You Will Not Find in React
- Motivation for Creating React
- A React JavaScript Example
- One-Way Data Flow
- JSX
- A JSX Example
- The Virtual (Mock) DOM
- Only Sub-components that Actually Change are Re-Rendered

- React Libraries
- Summary

2 - ES6 in a Nutshell

- What is ES6?
- ES6 Features
- Using ES6
- Transpiling
- Major Syntax Changes
- let and const
- Variable Scope
- Shadowing Variables
- Arrow Functions
- Arrow Functions As Parameters
- Using 'this' Within Arrow Functions
- Template Literals
- Spread Operator
- ES6 Classes
- Declaring Classes
- Declaring Instance Methods
- Accessor Methods
- Static Methods
- Inheritance With Classes
- Summary

3 - Babel Command-Line Interface

- Babel Transpiler
- Usage Options
- Presets and Plug-ins
- Babel CLI Installation
- Babel Configuration
- Running Babel Command-Line
- A Basic ES6 Development Setup with Babel
- Test the Babel Development Setup
- Adding React to the Development Setup
- Create a Minimal React App - Index.html
- Create a Minimal React App - app.js
- Summary

4 - Basic Components and JSX

- What is JSX?
- JSX Transpilation to React Code Example
- Running the Transpiled Code
- Babel
- The Babel Runtime JavaScript Library
- Script Import Skeleton Code
- Playing Around in CodePen
- React Components
- Ways to Create UI Components
- Creating a Functional Component Example
- Component Names Must Be Capitalized
- Creating a React Class-Based Component in ES5
- The render Method
- Creating a UI Component Using ES6 Class Notation
- Using ES6 Classes with React
- Which UI Component Creation Syntax Should I Use?
- Components vs Elements
- Elements Are Immutable
- Properties
- Property Naming Convention
- Properties Default to 'True'
- Spread Attributes (an ES6 Feature)
- Expressions
- Fragments
- Summary

5 - React Functional Component Concepts

- Functional Components
- Nesting JSX Elements
- Example of JSX Nesting
- Comments in JSX Code
- Setting CSS Styles Using Classes
- Setting CSS Styles Directly
- JSX Escapes Values
- Working with Lists of Items
- Keys in Lists
- Example List With Key
- Container vs. Presentational Components
- State
- Types of State Data

- State Hierarchy
- Lifting State Up
- Props vs. State
- Pass Down a Function
- Immutability
- Immutability – Why?
- Virtual DOM and State
- Setting state
- Updating Input fields
- Passing Props to Components
- Passing Functions to Components
- Event Handling
- Event Handler Example
- Event Binding - DOs
- Event Binding – Don'ts
- Passing Parameters to Event Handlers
- Component Life-cycle
- Life-cycle in Functional Components
- App Development Workflow – 1/3
- App Development Workflow – 2/3
- App Development Workflow – 3/3
- Summary

6 - React Components with ES6 Classes

- Classes in ES6
- Functional Components
- Extending React.Component
- The render() Method
- state
- props
- defaultProps
- propTypes
- Component Lifecycle
- Component Life-cycle: Overview
- Component Life-cycle – Render Phase
- Component Life cycle – Commit Phase
- Component Life-cycle – Unmounting
- constructor() example
- componentDidMount() example
- setState(newStateValue)
- Summary

7 - React Router

- Routing and Navigation
- react-router
- Creating a react-router based project
- A Basic Routed Component
- Router vs. BrowserRouter
- The Route component
- <Switch>
- Redirect Route
- Navigating with <Link>
- Navigating with <NavLink>
- Route Parameters
- Retrieving Route Parameters
- QueryString Parameters
- Using Router with Redux
- Summary

8 - State Management for React

- React State Basics – Props and State
- Props
- State in Class Based Components
- Managing State with Hooks in Functional Components
- The Problem with Props and State
- Redux State Library
- Redux Advantages
- Redux Disadvantages
- Basic Rules for State Management
- Types of State
- Data State
- Communication State
- Control State
- Session State
- Location State
- Location State Side Effects
- Summary

9 - Using React Hooks

- Functional Component Shortcomings
- Hooks Overview
- Hook Rules
- React Linter Example
- Functional Component Props
- The useState Hook
- Functional Component using the useState hook
- useState with Multiple Variables
- useState can also be used with Objects
- The useEffect Hook
- useEffect Hook Example
- Using useEffect Hook to Load Data
- Restricting when useEffect is Called
- The useContext Hook
- Additional Hooks
- The useReducer Hook
- An Example Reducer Function
- Calling and Using useReducer
- The useMemo Hook
- useMemo Example
- The useCallback Hook
- useCallback Example
- The useRef Hook
- Using useRef to Hold Values
- The useImperativeHandle Hook
- useImperativeHandle Hook Example
- The useLayoutEffect Hook
- Summary

10 - Unit Testing React with React Testing Library

- React Testing Framework
- Features
- Snapshot Testing
- Code Coverage
- Interactive Mode
- Projects created with create-react-app
- Default App Component Test
- Unit Tests
- Anatomy of a Unit Test

- Common Matchers
- Combining Tests
- Running Tests
- Testing Promise based async code with 'done'
- Setup and Teardown
- react-testing-library
- A Simple Component Test
- A Simple Snapshot Test
- Running and Updating SnapShot Tests
- Building Component Tests
- Calling Render
- Render Properties
- Simulating Events
- Testing Results
- Using Query Functions
- Text Matching
- Counter Component
- counter-test.js
- Summary

11 - Introduction to MongoDB

- MongoDB
- MongoDB Features
- MongoDB on the Web
- Positioning of MongoDB
- MongoDB Applications
- MongoDB Data Model
- MongoDB Limitations
- MongoDB Use Cases
- MongoDB Query Language (QL)
- The CRUD Operations
- The find Method
- The findOne Method
- A MongoDB Query Language (QL) Example
- Inserts
- MongoDB vs Apache CouchDB
- Summary

12 - Working with Data in MongoDB

- Reading Data in MongoDB

- The Query Interface
- Query Syntax is Driver-Specific
- Projections
- Query and Projection Operators
- MongoDB Query to SQL Select Comparison
- Cursors
- Cursor Expiration
- Writing Data in MongoDB
- An Insert Operation Example
- The Update Operation
- Update Operation Options
- An Update Operation Example
- A Remove Operation Example
- Limiting Return Data
- Data Sorting
- Aggregating Data
- Aggregation Stages
- Accumulators
- An Example of an Aggregation Pipe-line
- Map-Reduce
- Summary

13 - Introduction to Node.js
- What Is Node.js?
- Applications of Node.js
- Installing Node.js and NPM
- "Hello, Node World!"
- How It Works
- Node.js is built on JavaScript: Benefits
- Traditional Server-Side I/O Model
- Disadvantages of the Traditional Approach
- Event-Driven, Non-Blocking I/O
- Concurrency
- Using Node Package Manager (NPM)
- The Express Server Framework
- Summary

14 - Basic Web Application Development
- Introduction to the HTTP Module
- The Request Handler Callback Function

- The Server Object
- Example Use of Server Object
- The Request Object
- The Response Object
- Parsing Request Body
- Serving Static Files
- The HTTP Client API
- Making POST/PUT/etc. Requests
- Where To go from Here?
- Summary

15 - Introduction to Express
- Introduction to Express
- Basic Routing Example
- Defining Routing Rules
- Route Path
- The Response Object
- Supplying URL Parameters
- Ordering of Routes
- Defining Catch All Route
- Full Example Web Service
- Summary

16 - Express Middleware
- Introduction to Express Middleware
- Writing a Middleware Function
- Binding to a Path
- Order of Execution
- Raising Error
- Handling Error
- Serving Static Files
- Handling POST Request Body
- Enable Response Compression
- Summary

17 - Accessing MongoDB from Node.js
- Getting Started
- The Connection URL

- Obtaining a Collection
- Inserting Documents
- Updating a Document
- Querying for Documents
- Deleting a Document
- Connection Pooling
- Summary

Building React Apps with Redux

- Redux
- Redux Terminology
- Redux Principles
- Redux: Actions
- Redux Action Types
- Action Creators
- Dispatching Actions
- Data Flow Basics
- Redux Reducers

- Pure Functions
- Reducer Example
- Returning Default State
- Creating a Development Environment with create-react-app
- Using Redux with React
- Initializing the Store
- Immutability
- Benefits of Immutable State
- Mutability of Standard types
- Copying Objects in JavaScript
- Copying Arrays in JavaScript
- One Store - Multiple Reducers
- Combining Reducers
- Components and Redux
- The React-Redux Package
- Wrapping App with Provider
- mapStateToProps
- mapDispatchToProps
- Using Mapped Properties and Methods
- Wrapping Components with Connect
- Configure Store
- Programming Advice - MultiTab Console
- Summary

Events in Node JS

- Event Driven Programming
- Event Driven Programming (Contd.)
- Event Emitter
- EventEmitter Class
- EventEmitter Class – Inheritance
- The Event Loop and Event Handler
- Phases Overview
- Event Handlers
- Example (Using EventEmitter as an Object)
- Example (Inheriting from EventEmitter)
- EventEmitter Functions
- Issue with 'this' Keyword in Callback Functions
- Handling this Problem
- Controlling Event Callbacks in the Event Loop
- Summary