**REACT, REACT ROUTER + REDUX**

**(REACT-201)**

**LEVEL:** BASIC TOINTERMEDIATE

**OVERVIEW**

3 days for seasoned programmers | 4 days for freshers

7.25 hours/day (excluding time for lunch break and 2 tea breaks)

React is a web front-end framework used to create user interfaces, i.e. a view in MVC architecture. It is a free and open-source framework created by Facebook and others. It is not a full-fledged Single Page Application (SPA) framework, and hence other libraries like React Router and Redux/Mobx are required to build an SPA. Additionally, a module bundler like Webpack is usually used in a React application.

React Router is a popular library for setting up routing (navigation between pages) in a React application.

Redux is a JavaScript state management library popularly used along with React.

**PREREQUISITES**

* Working knowledge of HTML, CSS
* Very good knowledge of JavaScript – functions, objects, closures and the DOM. However knowledge of ES2015+ (ES6+) features is NOT required.
* Bootstrap knowledge is a plus, but not required
* Knowledge of Object Oriented Programming (OOP) concepts is desirable, but not required

**APPLICATION BUILT DURING TRAINING**

**One of these projects can be chosen as the running case-study**

*Events application*

Add the end of this bootcamp, participants would have built a events application. They shall be provided a backend server. The application will involve communicating with the backend and listing events (workshops), adding, editing and removing events & sessions of events.

*Store application*

Add the end of this bootcamp, participants will build a product catalog application. They shall be provided a backend server. The application will involve communicating with the backend and listing products, adding, editing and removing products, posting product reviews etc.

**LIST OF SOFTWARE TO BE INSTALLED BEFORE TRAINING BEGINS**

1. Git CLI on participant systems and GitHub account should be created for every participant (to be created individually by participant). The **GitHub account should be a personal one** and not one associated with the company’s GitHub account (I will not be able to add a company account as collaborator on my repositories, and hence shall not be able to share code).

Git CLI download: <https://git-scm.com/downloads>

GitHub link for account creation: <https://github.com/join?source=header-home>

Open a terminal and check installation went on fine by typing

$> git --version

You will see the version number of git ($> indicates the command prompt)

Once an account is created by everyone, **the list of GitHub user names needs to be shared with me** - I will add them to the GitHub repository before start of training.

1. Node.js needs to be installed on all systems – Mac OSX, Linux and Windows is supported. The 10.x.x (LTS version) may be installed. This will also install npm. However, the proxy server details may need to be configured to enable npm access the npm registry (this registry is required to download Node modules required to build Node applications) – **the proxy configuration is a necessary step and has to be completed before training starts**.

Node.js <https://nodejs.org/en/download/>

To configure the proxy for npm these articles will be helpful:

<https://jjasonclark.com/how-to-setup-node-behind-web-proxy/>

<https://forum.freecodecamp.org/t/npm-behind-a-proxy-server/19386>

Open a terminal and check installation went on fine by typing

$> node -v

$> npm -v

You will see the version number of node and npm tools

1. Open a terminal and install the following

$> npm install -g typescript create-react-app

1. Download and install Visual Studio Code (VSCode) from <https://code.visualstudio.com/download>

It is available for Windows, Mac OSX and popular Linux distributions.

1. Latest version of Chrome. Internet Explorer is not acceptable.

Chrome: <https://www.google.com/chrome/browser/desktop/index.html>

1. **Additionally, it would be great if participants have as little restrictions (as permissible) on internet access during the session**

**CHAPTERS AND TOPICS**

**Features of ES2015+**

Block-level scoping and the use of let, const

Template strings

Default Parameters

Object and Array Destructuring

Rest and spread operators (includes object spread)

Arrow Functions

Classes, Class Inheritance

Modules

Promises

**Before beginning React…**

The Axios library for making Ajax calls (The fetch API can be covered instead)

The Single Page Application (SPA) architecture

Brief introduction to Webpack

**Introduction to React**

Overview of React, Redux and React Router

Component-based architecture for front-end apps

Getting started with React – including it in your application

Scaffolding a React application using boilerplate code (create-react-app)

Understanding the Project Structure and build process

React elements, props and state

Immutability of React elements including props

**Component Basics**

Introduction to Components in React

Function and class-based components

Example: Clock component

Updating content by replacing elements

Creating a Clock component

Taking inputs using props

Children of React elements

Composing components

**Using JSX**

React.createElement() vs document.createElement()

What is JSX?

Need for JSX

Passing various types of props

Variables and Expressions

Example: Invoice Component

Comments in JSX

Conditional expressions and hiding and showing elements conditionally

Rendering an array of React elements

Iterating through arrays to render array of React elements

Styling React elements

**Stateful Components in Depth**

What is state and when is it required for a component?

Updating component state using setState

Forcing updates

Lifecycle methods – Mounting, Update and Unmounting phases and their methods

Handling asynchronous operations during the lifetime of a component

Example: Countdown timer component

Parent-child upstream/downstream communication

Sending props, state, children etc. downstream

Communication from child to parent component using parent function passed as prop

Example: Collapsible panel component

Basics of event handling

Binding the context and arguments of event handlers

Event object properties and methods

Validating prop data type using PropTypes

Setting default values for props using defaultProps

Virtual DOM – DOM diffing and reconciliation

Setting a key for efficient DOM rendering

Using refs for fetching DOM node references

Working with forms and validating inputs - default value for input elements, controlled and uncontrolled components

Example: Store application with React

**Debugging**

Using browser debugger

Debugger statement

React devtools for state snapshots

**Routing**

Introduction to React Router

Example: Store application with React and React Router

Route configuration – Link, NavLink, Route, Switch, Redirect components

The history, location and match props

Handling params and querystring

**Redux**

The Flux architecture

Redux flow overview

Actions and Stores

Immutability

Reducers

Middleware in Redux and popular Redux middleware

Implementing custom middleware

Redux Thunk (redux-thunk)

Example: Store application with React, React Router and Redux

Connecting React to Redux (react-redux) – Provider, mapStateToProps & mapDispatchToProps

Redux dev tools

**Deployment**

Configuration management in a create-react-app application

Creating a production build

Deployment