**REACT, NEXT.JS WITH TYPESCRIPT, REACT ROUTER, REDUX, UNIT TESTING**

**(REACT-403)**

**LEVEL:** BASIC TOADVANCED

**OVERVIEW**

**5 full days** for **seasoned programmers with working knowledge of JavaScript, React**

**8 full days** for **seasoned programmers with working knowledge of JavaScript but not React**

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 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.

Next.JS is a React framework created by the Vercel team. It implements various techniques that enable creation of performant full-stack React-based applications, and is particularly useful when creating Server-Side Rendered (SSR) or Statically Generated Sites (SSG).

**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.
* Working knowledge of React
* Bootstrap knowledge is a plus, but not required
* Knowledge of Object Oriented Programming (OOP) concepts is desirable, but not required

**APPLICATION BUILT DURING TRAINING**

*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. The API access will be authenticated and the app shall have login/logout functionality.

**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)

**The list of GitHub user names needs to be shared with me**.

1. Node.js needs to be installed on all systems – Mac OSX, Linux and Windows is supported. The 20.x.x (LTS version) may be installed. This will also install npm.

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

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. 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**

**Select Topics in JavaScript (0.5 days / 1.5 days)**

This provides only a brief refresher, with no hands-on (instructor will code and demonstrate)

**Overview of features of ES2015+**

Block-level scoping and the use of let, const

Template strings

Object and Array Destructuring

Rest and spread operators (includes object spread)

Arrow Functions

Modules

Promises

async..await

**Overview of TypeScript**

Installation and getting started

The tsc compiler options and configuration using tsconfig.json file

Primitive types and the any type

Static type checking, type inference, type narrowing, type assertion

Arrays

Type Aliases

Union types

Defining function argument and return types

Function signatures involving callback functions

Using interface to define structure for an object (properties and methods)

Implementing interfaces in classes

Access modifiers – public, protected, private and readonly

Using Generics

Utility types – Omit, Partial, Pick, Required, Record

Mapped Type

Conditional Type

**React, React Router, Redux using Redux Toolkit (1.5 days / 2.5 days)**

This provides a very brief refresher, with only select hands-on (instructor will code and demonstrate, and share exercises at chosen points where participants can apply what was covered).

**Introduction**

The Single Page Application (SPA) architecture

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

**Component Basics**

Introduction to Components in React

Function and class-based components

Taking inputs using props

Children of React elements

Composing components

Need for JSX

Passing various types of props

Variables and Expressions

Conditional expressions and hiding and showing elements conditionally

Rendering an array of React elements using map()

Styling React elements

Basics of event handling

Binding the context and arguments of event handlers

Event object properties and methods

Setting default values for props using defaultProps

**Stateful Components in Depth**

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

useState()

Handling side-effects and asynchronous operations during the lifetime of a component

useEffect() and dependencies array

Parent-child upstream/downstream communication

Sending props, state, children etc. downstream

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

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

**Hooks and Performance Optimization**

What problems hooks solve

Handling complex state transitions using useReducer()

Performance optimization using useCallback(), memo() and useMemo()

Props drilling and avoiding it using the context API – createContext(), useContext(), Provider

Creating custom hooks

Container-vs-Presentation components, HOCs and why function components with custom hooks are preferable

Render props

React Query/TanStack Query for handling data fetching states, and data caching

**Routing**

There will not be much focus on this as Next JS has a different way of routing

Introduction to React Router

Example: Store application with React and React Router

Route configuration – Link, NavLink, Routes, Route, Navigate components

The history, location and match props

Handling params and querystring

Performance optimization using code-splitting, lazy, Suspense, and route-based code-splitting

Programmatic route changes

**Redux using Redux Toolkit, Basics of Zustand**

The Flux architecture

Redux flow overview

Actions and Stores

Immutability

Reducers

React Redux

Redux Toolkit for simpler setup of store

useDispatch() and useSelector()

Example: Store application with React, React Router and Redux

Redux dev tools

Basic introduction to Zustand for maintaining state

**Deployment**

Configuration management in a create-react-app application using .env files

Creating a production build

Deployment

**Next.js (3 days / 4 days)**

Since focus is on latest features, Next.JS v15 shall be used. However, working with Next.JS 13 (Pages router) shall also be briefly covered. Tailwind CSS will be used for the UI.

**Introduction**

Features of Next.js

Generating a Next.js application using create-next-app

Understanding the folder structure (Pages Router vs App Router)

Understanding the toolchain and build process

The next.config.js file and other config files

ESLint, Prettier, TypeScript setup

**Getting started with the application**

Example: Store application with Next.js

Pages and routing (Pages Router vs App Router)

Dynamic routing, using path parameters (Pages Router vs App Router)

Page, layout, template

Linking using next/link

useRouter() hook

Dynamic routes and the usePathname() hook

Nested layouts and child routing

Grouping routes

Including image using next/image and lazy loading images

Handling responsiveness, layout shift

CSS approaches – module CSS, Sass

Global styles

Fonts and font optimization using next/font

Including scripts and static assets

SEO and metadata (Pages Router vs App Router)

**Interacting with the backend (Pages Router and App Router)**

API routes (Pages Router vs App Router)

NextRequest and NextResponse

Handlers in v13 and v14

Form validation on the server-side using Zod

Making API calls from the client-side

Redirection, cookies

**Rendering models (Pages Router and App Router)**

Build-time and request time rendering

Server-Side Rendering (SSR), advantages and use-cases

getServerSideProps() (Pages router)

How SSR works at request time and later

Static Site Generation (SSG), advantages and use-cases

getStaticProps(), getStaticPaths() (Pages Router)

How SSG works at build time and later

Fallback and revalidation

Incremental Static Regeneration (ISR)

The older getInitialProps() API

Differences in rendering behavior in development and production modes

Lazy loading using next/dynamic

How caching works in Next.js

**New Rendering models (App Router)**

Server and Client components

Advantages of Server components

How rendering happens

Static and dynamic rendering

Designing component hierarchy with the new rendering model

The useFormStatus() hook

Streaming and partial rendering

Server actions and mutations

Data fetching patterns – on client, on server, handle caching and revalidation

**Authentication and Authorization**

Setting up token-based Authentication

Protecting Routes using middleware

Protecting Server Actions

Authorization using Server Components

**Unit Testing and Deployment**

Setting up for unit testing using Jest and RTL

Handling static file imports during tests

Writing and running tests

Testing components

Testing API routes

Preparing the production build

Analyzing the bundle using @next/bundle-analyzer plugin

Production build and the .next folder

Deployment

Copilot Usage and Best Practices