Prudentia

REACT, NEXT.JS WITH TYPESCRIPT, REACT ROUTER, REDUX
(REACT-403)

LEVEL: BASIC TO ADVANCED

OVERVIEW

40 hours for **seasoned programmers with working knowledge of HTML, CSS and JavaScript**

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
- Good knowledge of JavaScript – looping, branching, arrays, functions, objects
- 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.

Prudentia

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.

2. Node.js needs to be installed on all systems – Mac OSX, Linux and Windows is supported. The 22.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

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

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

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

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

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

Prudentia

CHAPTERS AND TOPICS

Select Topics in HTML, CSS, JavaScript (4 hours)

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

HTML and CSS

Flex box layout

Grid layout

Responsive Web Design Concepts

JavaScript

Functions as first-class citizens - Passing functions as arguments

Object and Array Destructuring

Rest and spread operators (includes object spread)

Arrow Functions

Modules

Promises

Event loop – asynchronous programming

async..await

UI/UX for Web Developers

Use of whitespace, alignment, and typography.

Grids and consistent spacing.

Practical tips using CSS, Tailwind, or component libraries

Designing for various screen sizes from the start

Media queries, flex/grid techniques.

Visual cues: hover/focus/disabled states, loading indicators

Accessibility best practices: semantic HTML, keyboard nav, ARIA.

Tools: Lighthouse, React ARIA

TypeScript, React, React Router, Redux using Redux Toolkit (16 hours)

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).

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 and type inference
- 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
- Using Generics

Introduction to React

- 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

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

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

Deployment

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

Creating a production build

Deployment

Next.js (20 hours)

Since focus is on latest features, Next.JS v14 shall be used. However, working with Next.JS 13 shall also be briefly covered.

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

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
- 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

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

Preparing the production build

Analyzing the bundle using @next/bundle-analyzer plugin

Production build and the .next folder

Deployment