Trainer’s name: [Banu] [Prakash]

Program objective:

Web application development using ReactJS using JavaScript

Pre-requisite:

For this tutorial one should have a good working knowledge of HTML and CSS.

Duration of program

6 Days

H/w – S/w required for Lab Setup

Type of hardware	List of software
NA	**Operating System: Windows/MAC/ Linux/ Unix** **Web Browsers: Chrome, Firefox and IE.** **Editors: Sublime Text / Visual Studio Code.**

Target Audience

This tutorial is designed for the aspiring Web Designers and Developers

Day 1:

JavaScript

Introduction to JavaScript

- JavaScript language fundamentals
- The Flexibility of JavaScript
- JavaScript as a Loosely Typed Language
- Inline JS.

Functions

- Creating functions
- Calling functions
- Returning values
- Anonymous Functions
- Function Literal
- Function Invocation Patterns
- Functions as Callbacks

OOP with JavaScript

- Understand JavaScript Object notation (JSON)
- Writing function constructor
- Writing class owned instance methods
- Understand prototype for writing object owned instance methods
- Writing class owned class methods
- Understand Prototypal inheritance
- Using Array Objects

Functional style of Programming

- Functions as First-Class Objects
- Closures
- High Order functions
- Writing high order functions
- Currying
- Implementing Map, reduce and filter functionalities using high order functions

ES2015

- Arrow functions – A short-hand version of an anonymous function.
- Block-level scope – ES6 now supports scoping variables to blocks (if, for, while, etc.) using the let keyword.

- Classes – ES6 classes provide a way to encapsulate and extend code.
- Constants – You can now define constants in ES6 code using the const keyword.
- Default parameters – Ever wished that a function parameter could be assigned a default value? You can do that now in ES6.
- Modules – Provides a modular way of organizing and loading code.
- Promises – Used with async operations

Day 2:

Handling Events

- DOM
    - Creating DOM elements
    - Accessing DOM elements
    - Handling events

JQuery

- Selecting and Manipulating DOM Elements with jQuery
    - Using CSS Selectors
        - Basic CSS Selectors
        - Hierarchical Selectors
        - Attribute Selectors
        - Adding jQuery Filters to Your Selectors
- Manipulating the Document Object Model (DOM) for Cross-Browser DHTML
    - Leveraging the .ready() method
    - Adding and replacing content with jQuery
    - Updating, adding and deleting element content
    - Inserting nodes into the DOM and manipulating parents and siblings
- Dynamically assigning CSS properties
    - Adding and removing CSS rules and classes
    - Controlling element size and position
- jQuery and Ajax
    - The load() Method
    - Basic Ajax Requests with $.get() and $.post()
    - $.getJSON() and $.getScript()
    - Exercising Complete Control with $.ajax()
    - Global Ajax Events
    - Ajax Helper Methods

Day 3:

Node.js

- Introduction to NodeJS
- NPM

- Writing asynchronous code
- Modularizing code
  - Understanding built-in modules
  - Techniques for modularizing JavaScript code
  - Using require() to modularize application code
  - Using npm for third-party modules
- Handling Exceptions
- HTTP Server with Node.js and Core http Module
  - Node.js, Web Apps and http Core Module
  - Node.js Hello World HTTP Server
  - Node.js Hello World HTTP Server Demo
- **JavaScript Unit testing**
  - Mocha and Chai

JavaScript build tools

- Webpack
  - Static module bundler
  - Write webpack.config.js
    - Entry
    - Output
    - Loaders
    - Plugins
    - Setting Mode
      - Production
        - Optimized, minimized, source-mapped bundle
      - Development
        - webpack-dev-server for hot-reloading, debugging enabled

Day 4 and 5:

ReactJS

ReactJS

Introduction

- What is React?
- Real World SPAs & React Web Apps
- React Alternatives

React Components

- Component basics
- Component architecture
- Virtual DOM
- Splitting app into components
- Functional components

- Component Implementation
- Component Composition
- Composition Implementation
- Lifecycle Methods
- JSX
- React State and Props
    - Manging Data in React
    - State and Props Implementation
- React Event Handling
- Working with Forms and Events

Testing React

- Introduction to JEST
- React Testing Library
- Rendering a component
- Selecting elements
- Search Types
        - getByText, getByRole, getByLabelText, getByPlaceholderText, getByAltText, getByDisplayValue
        - Search Variants: queryByXXX and findByXXX
        - Using **container** to query for rendered elements
- Using screen.debug()
- Using RTL's Assertive functions
- Fire Event and User Event
- Mocking callbacks and Testing async
- Code coverage
- E2E testing with Cypress

Styling

- Using styled components

React-Router

- Routing and SPAs
- Setting up links
- Rendering components for Routes
- Using Routing related props
- Passing and extracting Route parameters
- Navigating programmatically
- Redirecting Requests

Context

- Passing data through the component tree without having to pass props down manually at every level
- React.createContext
- Context.Provider
- Context.Consumer

HTTP/ Connecting to REST endpoints

- Fetching data via Ajax
- Rendering fetched data to the screen
- Posting data via Ajax
- Creating and using Axios / fetch

High Order Components

- Props proxy
- Inheritance Inversion

Error boundaries

- Use static getDerivedStateFromError() to render a fallback UI after an error has been thrown.
- Use componentDidCatch() to log error information.

Day 6:

Refs and DOM

- Creating Refs
- Forwarding Refs

React Hooks

- useState
- useReducer
- useEffect
- useCallback
- useMemo
- useRef
- useContext

Redux

- Problems of Flux pattern
- Building blocks in Redux
- Action

- Action Creators
- The store
- The reducers
- Combine reducers
- Views: smart and dumb view
- React-Redux Bindings
- The root component
- The data flow in Redux

Middleware

- Using Middleware
- Creating Custom Middleware
- Creating a Logger Middleware
- Configure Redux DevTools Extension
- Redux Thunk
    - Handling Asynchronous Redux actions
- Redux-Saga
    - Make application side effects easier to handle
    - Using Redux saga helper functions: takeEvery(), takeLatest(), put(), call()
    - Running effects in parallel: all() and race()
- Using Redux Toolkit

React’s Performance

- The shouldComponentUpdate()
- PureComponent
- React memo
- Binding in Constructors vs Arrow functions
- Avoid binding when rendering
- Using proper key property while rendering lists
- React Fragments
- Debouncing event action