Course duration

- 5 days

Course Benefits

Course Outline

1. Setup
    1. Verifying Node.js and either NPM or yarn
    2. Verifying class libraries
    3. Verifying class files
    4. IDE (WebStorm or Visual Studio Code preferred)
2. Introduction to React
    1. What problem(s) does React solve?
        1. Traditional, pre-JS web applications
        2. Late-model, MV* and JS web applications
    2. React's solutions
        1. Single-page apps
        2. View libraries
        3. Helper libraries
    3. React development environment
        1. Simplicity: create-react-app
        2. Build-your-own: an overview
    4. Hello world
        1. Your first React component
        2. Using React within a page
        3. Making some basic changes
        4. React and JSX
3. Components
    1. Two types of components
        1. Functional components
        2. Class-based components
        3. Why use one or the other?
    2. Testing basic components
        1. Testing libraries: Enzyme vs Testing Library (sic)
        2. Jest
        3. Testing with Testing Library
    3. Props and state
        1. Passing in properties
        2. Limitations of properties
        3. Using state and the useState() hook

        4. When to use state, when to use props
        5. Testing state and prop changes
    4. Event handling
        1. React event handling
        2. Synthetic events
        3. React vs DOM event handling
        4. Testing events
    5. Children
        1. Components within components
        2. Known children and unknown children
        3. Testing child components
    6. Parent-child component communication
        1. Communication from parent to child
        2. Communication from child to parent
        3. Container vs presentational components
4. React Component Lifecycle
    1. Lifecycle overview
        1. Startup and mounting
        2. Rendering
        3. Updating
        4. Unmounting
    2. Using useEffect() for lifecycle methods
        1. Run once
        2. Run every render
        3. Run on specific changes / updates
    3. Lifecycle methods in tests
    4. Error handling and error boundaries
5. Intermediate component usage
    1. PropTypes
        1. Typing and React
        2. Using PropTypes
        3. PropTypes in production
        4. PropTypes vs TypeScript
    2. Asynchronous data
        1. When should asynchronous fetching be done?
        2. What challenges does async offer?
        3. Asynchronous best practices
        4. Testing against async fetches
    3. Lists of data
        1. Iterating over a list
        2. The key property
        3. Sorting data
    4. Testing component interactions
6. Forms
    1. Controlled vs uncontrolled components
        1. What does React know about your form field?
        2. Does React control your form field?

        3. When does React find out about changes to your form field?
    2. Form field types
        1. Controlling a text field
        2. ther form fields
    3. Getting data out of a form
    4. Working with form data in tests
7. Introduction to Redux
    1. What problems does Redux solve?
        1. How does it solve them?
    2. Basic Redux pattern
        1. Store
        2. Reducers
        3. Actions
    3. A basic Redux example
8. Modern Redux with the Redux Toolkit
    1. What is the Redux toolkit
    2. What does it provide?
    3. The ducks pattern
    4. Testing Redux
9. React and Redux
    1. Plugging into React
        1. State as props
        2. Events as dispatch
        3. Introducing higher-order components
    2. Turning our standalone Redux program into a component
    3. Middleware
        1. Provided by the toolkit
        2. ther middleware
    4. Building a real-world React-Redux component
    5. Testing React-Redux components
    6. Higher-order components in detail
        1. What do higher-order components do?
        2. Why would I use a higher-order component?
10. Asynchronous Redux
    1. The difficulties of asynchronous Redux
    2. Asynchronous middleware
        1. Depending on your needs, we can use either thunks, sagas, or survey both techniques for asynchronous interactions
    3. Dispatching async actions
    4. Catching results
    5. Handling errors
    6. Testing asynchronous Redux

1. Setup
    1. Verifying Node.js and either NPM or yarn
    2. Verifying class libraries
    3. Verifying class files

4. IDE (WebStorm or Visual Studio Code preferred)

2. Context
    1. What is the Context API?
    2. How to use the Context API
        1. Hooks-based Context: the useContext() hook
        2. Class-based Context
    3. Testing while using Context
3. Advanced Redux
    1. Using reselect to minimize re-rendering
    2. Normalizing state
    3. Higher-order reducers
    4. Helper libraries
4. Introduction to routing
    1. What problem is routing trying to solve?
    2. How does routing solve this problem?
        1. Tying components to URLs
        2. Passing parameters via the URL
    3. Routing software: React Router
    4. Simple router example
    5. Testing routing
        1. More complex routing
    6. Top-level routing
        1. Routing at the top of your application
        2. Allowing other parts of the application to manage routing
    7. Redirects
    8. React-router objects
        1. match
        2. history
        3. location
    9. Routing organizational techniques
    10. Testing advanced routing
5. Advanced React
    1. Understanding and optimizing reconciliation
        1. Best practices for React reconciliation
        2. Recognizing common issues
        3. Making improvements
    2. Refs
        1. What's a ref?
        2. What problem does it solve?
        3. How can I use refs?
            1. Hooks: the useRef() hook
            2. Classes and createRef()
        4. The challenges of testing refs
6. Render props
    1. Rendering in depth
    2. Rendering a function instead of a prop
    3. Using the render prop pattern

4. Testing render props

Class Materials

Each student will receive a comprehensive set of materials, including course notes and all the class examples.

Class Prerequisites

Experience in the following *is required* for this JavaScript class:

- 1-2 years of JavaScript experience.
- Advanced understanding of JavaScript, including prototypes and functions as first class citizens.