Course duration

- 3 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. Verifying TypeScript setup
    5. 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 and TypeScript development environment
        1. Simplicity: create-react-app with TypeScript built in
    4. Hello world
        1. Your first React component
        2. Using React within a page
        3. Making some basic changes
        4. React and JSX and TypeScript
3. Components
    1. Two types of components
        1. Functional components
            1. Functional component types
        2. Class-based components
            1. Class component types
        3. Why use one or the other?
            1. Important distinctions before version 16.8
                1. Class-based components for state and lifecycle
                2. Functional components for simplicity and purity
        4. Currently, prefer functional components with hooks
    2. Testing basic components
        1. Testing libraries: Enzyme vs Testing Library (sic)

        2. Jest
        3. Testing with Testing Library
        4. Testing with TypeScript
    3. Props and state
        1. Properties and types
        2. Passing in properties
        3. Limitations of properties
        4. State and types
        5. Using state and the useState() hook
        6. When to use state, when to use props
        7. Testing state and prop changes
    4. Event handling
        1. React event handling
        2. Event types
        3. Synthetic events
        4. React vs DOM event handling
        5. Testing events
    5. Children
        1. Component types
        2. Components within components
        3. Known children and unknown children
        4. 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. Using types to validate communication
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. Asynchronous dat
        1. When should asynchronous fetching be done?
        2. What challenges does async offer?
        3. Working with Promises and generic types
        4. Asynchronous best practices
        5. Testing against async fetches
    2. 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. Form field types
        2. What does React know about your form field?
        3. Does React control your form field?
        4. 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?
    2. How does it solve them?
    3. Basic Redux pattern
        1. Store
        2. Reducers
        3. Actions
    4. Redux types
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. Types with React-Redux
        1. Too many variations
        2. Using Generics
        3. Solving TypeScript issues with React-Redux
    3. Turning our standalone Redux program into a component
    4. Middleware
        1. Provided by the toolkit
        2. ther middleware
    5. Building a real-world React-Redux component
    6. Testing React-Redux components
    7. 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
    2. Types as appropriate
3. Dispatching async actions
4. Typing async results
5. Catching results
6. Handling errors
7. Testing asynchronous Redux

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.