STRONGBYTES

ADVANCED REACT

As React grew in popularity, so did the number of libraries and utilities meant to integrate with it. Redux is the most well-known implementation of a one-way dataflow architecture. By embracing functional programming concepts, promoting simplicity and exposing a very simple API, Redux quickly became a popular choice for managing state within React projects. On the other side, React Router dominates the routing scene and has become the de-facto solution for client-side routing in React apps.

In this training we will dissect and understand what one-way dataflow architecture is. We will grasp the need of a state management library and explore the advantages it brings. We will fully integrate Redux, build our own utility codebase for state management using the Context API, use React Router and briefly work with component libraries (Material UI). We will also talk about performance tips in order to avoid render storms and best practices about how to construct your global store. Feel free to consult the high-level agenda below.

INTENDED AUDIENCE

This training assumes the participants are already familiar with ES6 concepts (classes, destructuring, default parameters, arrow functions, let/const, etc.), basic JavaScript knowledge and syntax, npm (interacting with the npm ecosystem, package.json), Git (basic commands) and that they have basic React knowledge (JSX, VDOM, state, props, event management, component lifecycle hooks, etc.). Thus, the aforementioned concepts will not be discussed or detailed during the training.

This training targets intermediate developers who would like to upgrade their React knowledge by finding out more about and trying technologies such as Redux, React Router and implementing performance best practices in UIs built upon this technical stack.

OUTLINE

Included below are the main topics covered in this training. The duration is 3 days.

1. State in big applications
    a. Architectural patterns - lifting state up for sharing information
    b. Issues with state lifting in complex applications
2. What is Redux?
    a. Introduction to Redux
    b. What problems does Redux solve?
    c. Redux as an architectural pattern in the JavaScript ecosystem
3. Redux basics
    a. Concept of a pure function (small dive into functional programming)
    b. Actions, Reducers, Store
    c. The overall data flow in a React + Redux app

4. General architecture of the puppy adoption app
    a. Discussing the high-level approach to building a puppy adoption app with React and Redux
5. Actions
    a. What is an action?
    b. Simple actions vs. action creators
    c. Creating and dispatching an action
    d. Async actions and redux-thunk
    e. Testing the actions / action creators
6. Reducers
    a. What is a reducer?
    b. Creating a reducer & testing it
    c. Combining multiple reducers (talk about react-redux and combineReducers)
7. Small introduction to HOCs
    a. What is a HOC?
    b. How can HOCs be useful?
    c. Example of HOCs in the wild

8. Store
    a. What is the store?
    b. Connecting a React component to the store
    c. Using react-redux inside the application
    d. mapStateToProps and mapDispatchToProps explanation
    e. Closing the loop on the data flow
9. Implementing the puppy adoption application
    a. Listing all the puppies
    b. Implement filtering
    c. Implement CRUD operations
10. Building our own state management library
    a. Context API and hooks
    b. Build a simple way to manage data flow using Context API and useState, useEffect and useReducer
    c. Introduction to simpply
11. React Router
    a. Introduction to Single Page Apps (SPAs)
    b. React Router basics

c. Parameterized routes, query data, redirects

d. Accessing route information anywhere via withRouter

e. Adding the individual puppy view in the application

12. Component libraries

a. What are component libraries and why are they useful?

b. Examples of component libraries

c. Integrating Material UI components in the puppy adoption app

13. Performance in React

a. An overview of the reconciliation algorithm

b. Using keys in React

c. Performance considerations of hooks and functional components

d. Build for production

e. React.memo, React.lazy and Suspense

f. Avoiding render storms (nested state best practices)

g. Profiling your React app and identifying performance bottlenecks

14. Tweaking and improving the puppy adoption application

a. Identify potential performance issues

    b. Codebase overhaul

15. Recap

ABOUT THE TRAINER

Vlad Zelinschi is a pragmatic software architect, lover of the web and caffeine addict. He cares about surrounding himself with thoughtful leaders. He's the CTO of Strongbytes, Google Developer Expert on Web Technologies since October 2016 and a certified trainer. He's part of JSHeroes Community and actively acting as an advisor for a couple of well known Romanian conferences such as Codecamp and NDR.