Week 2 Workshop

React Native Course

Agenda

Activity	Time
Get Prepared: Log in to Nucamp Learning Portal • Slack • Screenshare	10 minutes
Check-In	10 minutes
Week Recap	30 minutes
Task 1 & 2	70 minutes
BREAK	15 minutes
Task 2 & 3	90 minutes
Check-Out	15 minutes

Check-In

- How was this week? Any particular challenges or accomplishments?
- Did you understand the Exercises and were you able to complete them?
- You must complete all Exercises before beginning the Workshop Assignment.

Welcome to React Native!

Week 2 Recap – Overview
Concepts Covered This Week

• Icons • Redux • Debugging	• Activity Indicator • Forms • Modals

- Only 30 minutes allotted to Week 2 Recap – the Workshop Assignment this week is a long one!
- Set an alarm now for 30 minutes and switch to working on the assignment no later than that time.

Review: Object Destructuring

- In the first exercise of Week 1 (Icons, Favorites, and Comments), we changed this line in CampsiteInfoComponent.js:

```
function RenderCampsite({campsite}) {
```

- To these two lines:

```
function RenderCampsite(props) {

    const {campsite} = props;
```

Review Question 1: Why was this change necessary?

Review Question 2: If we had not added the second line of `const {campsite} = props;`, what other code would we have had to change instead in the RenderCampsite component?

Advance to next slide after discussing these questions together.

Review: Object Destructuring (cont)

- Answer to Review Question 1:
  - This change was required because the newly added Icon component required access to props.favorite and props.markFavorite(), and the destructuring syntax only gave access to props.campsite.
- Answer to Review Question 2:
  - All instances of campsite would need to be changed to props.campsite.

Review: Setting up Redux

- Setting up Redux is more or less the same as with React (or any other JavaScript project)
- You installed redux, react-redux, redux-thunk, and redux-logger to your project
- You set up a baseUrl.js file with your computer's IP address and started using json-server again
- You defined action types and action creators
- You set up your reducer files – campsites.js, etc
- You created a Redux store with createStore() and combineReducers(), as well as applyMiddleware to bring in thunk and logger
- This should have all looked very familiar!

baseUrl.js and json-server

- If you are at an in-person workshop, or if you have changed locations from your weekday location:
  - Check right now for your computer's current IP address.
  - If you are on a different network than during the week, your IP address is most likely different.
  - You will need to change the IP address in your baseUrl.js file just for during the workshop (or any time that you are connected to a different LAN/your IP address changes).
- Start your json-server now, in preparation for your workshop assignment.
  - From within the folder that contains the correct db.json file, use the command **json-server -H 0.0.0.0 --watch db.json -p 3001 -d 2000**

Review: Redux in React Native

You updated the Redux Native code to use Redux, by:

- Wrapping the render of your App component with <Provider store={store}>
- Adding mapStateToProps() and connect() function to all the components to subscribe them to the Redux store and receive specific props
- Adding mapDispatchToProps object to Main component to access dispatch to actions
- Changing data source to Redux (which in turn fetches the data from json-server) instead of the files in the local shared folder
- Again, this is more or less the same as what you did in React, and any React or React Native project you use with Redux for state management is likely to follow a similar workflow

Review: Redux in React Native & function parameters in JavaScript

- In MainComponent.js, we saw the use of the connect() function with a null argument:

```
export default connect(null, mapDispatchToProps)(Main);
```

- We've also seen that we can use the connect() function with just mapStateToProps, ex.:

```
export default connect(mapStateToProps)(Home);
```

- Discuss together: Why can we *not* write the connect() function in MainComponent.js like this?:

```
export default connect(mapDispatchToProps)(Main);
```

Review: Redux in React Native & function parameters in JavaScript

- Answer to discussion question: The order of the parameters in a function's parameter list is important. Even though they are both optional, the mapStateToProps and matchDispatchToProps arguments must be given in a particular order, first and second.
- It's not the name of the argument that matters, what matters is its position in the parameter list. Thus if you tried this:

```
export default connect(mapDispatchToProps)(Main);
```

- The connect() function would try to treat it as if it was a mapStateToProps argument, and you would get an error.
- Thus, in this and any JavaScript function parameter list, remember that you must pass an argument of some kind, even if it's optional, to a preceding parameter if you're trying to pass in an argument to a later place on the parameter list.

Review: Debugging

- Make sure every student is able to access the Expo Developer Menu, show console in browser developer console by debugging remotely, and install/run/connect to react-devtools:
- Access the Expo Developer Menu on the emulator by using Ctrl-M or Cmd-M (or for mobile devices, shake the device slightly from side to side)
- If Ctrl/Cmd-M does not work for the emulator, open a bash terminal to any directory and type: adb shell input keyevent 82
- Choose Debug Remote JS from the Expo Developer Menu to see output from app to browser developer console (be sure to turn it off afterward)
- Run standalone react-devtools from bash terminal - may need to reload emulator to connect after starting react-devtools
- Choose Show Element Inspector from Expo Developer Menu then select components in application UI to inspect them inside react-devtools components hierarchy view

Review: arr.includes() method

- In the render method of the CampsiteInfo component, we initially passed the favorite prop to the RenderCampsite component in this way:

```
<RenderCampsite campsite={campsite}
    favorite={this.state.favorite}
```

- In the Redux Adding Favorites exercise, we changed this to:

```
<RenderCampsite campsite={campsite}
    favorite={this.props.favorites.includes(campsiteId)}
```

- In the first of the ways above, the favorite prop could be potentially passed with a value of true or false. **Discuss together:** What are the potential values of the favorite prop in the second way and why?

Review: arr.includes() method (cont)

- **Answer to discussion question:** The potential values are still true or false, because the *arr*.includes() method has a return value of true or false.
- The way you obtain that true or false value is different – before, you obtained it from the CampsiteInfo component's local state.
- Now you are obtaining it from the Redux store as an array of campsite IDs, then checking if a particular campsite's ID is (true), or is not (false), in that array.
- So you are still passing true or false as a prop to the RenderCampsite component. That's why you did not need to change anything in the RenderCampsite component. It's still receiving a Boolean value as it did before.

Forms

- Unlike in web development, there is no HTML <form> element that surrounds form inputs and handles the form submission with a form action. More freeform - you build it with JavaScript and React Native components.
- Use components from the React Native built-in components, e.g. Picker, Switch, TextInput, as well as from third party library components, e.g. Input from RNE, DateTimePicker from @react-native-community, many more options
- Typically store form values in local component state (controlled form)
  - Use a prop on the component such as onValueChange for Picker to change the state with setState when user updates the value
  - Set up a Button or other UI component that lets user submit the form, and set up code to handle the form submission

Workshop Assignment

- It's time to start the workshop assignment!
- Sit near your workshop partner.
  - Your instructor may assign partners, or have you choose.
- Work closely with each other.
  - 10-minute rule does *not* apply to talking to your partner, you should consult each other throughout.
- Follow the workshop instructions very closely
  - both the video and written instructions.
- Talk to your instructor if any of the instructions are unclear to you.

Check-Out

- Submit your 4 assignment files in the learning portal:
  - *ActionTypes.js*
  - *ActionCreator.js*
  - *comments.js*
  - *CampsiteInfoComponent.js*
- Wrap up – Retrospective
  - What went well
  - What could improve
  - Action items
- Start Week 3 or work on your Portfolio Project