React Native:

Week 2 Workshop Presentation

Workshop Agenda

Activity	Estimated Duration
Set Up & Check-In	10 mins
Week 2 Review	50 mins
Assignment	60 mins
Break	15 mins
Assignment	90 mins
Check-Out (Feedback & Wrap-Up)	15 mins

Set up

Along with Zoom, please also be logged into Slack and the learning portal!

Check-in

Check-In

How was this week? Any 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.

Week 2 Review

Overview

Review: Object destructuring	Array.includes()
baseUrl.js and json-server	Building forms
Redux in React Native	Handling forms

- 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 2 (Icons, Favorites, and Comments), we changed this line in **RenderCampsite.js:**

```
const RenderCampsite = ({ campsite }) => {
```

- To these two lines:

```
const 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

- Answer to Review Question 1:
  - This change was required because the newly added **Icon** component required access to **props.isFavorite** 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**.

baseUrl.js and json-server

- If you have changed locations from the last time you worked on your project:
  - **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: 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**, and **@reduxjs/toolkit** to your project
- You set up a **baseUrl.js** file with your computer's IP address and started using **json-server** again
- You set up your **slice reducer** files – campsitesSlice.js, etc
- You created a Redux **store** with the **configureStore()** method from **@reduxjs/toolkit**
- This should have all looked very familiar!

Review: Redux in React Native

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

- Wrapping the render of your **App** component with **<Provider store={store}>**
- Using the **useSelector()** hook to *view* items in the store
- Using the **useDispatch()** hook to dispatch actions to the store
- 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 how your app was structured with Redux in React, and any React or React Native project you use with Redux for state management is likely to follow a similar workflow

Array.includes()

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

```
<RenderCampsite
    campsite={campsite}
    isFavorite={favorite}
```

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

```
<RenderCampsite
    campsite={campsite}
    isFavorite={favorites.includes(campsite.id)}
```

- 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?

Array.includes()

- **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 **CampsiteInfoScreen'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.

Building forms

- Unlike in web development, there is no HTML **<form>** element that surrounds form inputs and handles the form submission with a form action.
- React Native approach is more freeform - you build it with JavaScript and React Native components.
- Use components from the React Native built-in components, e.g. **Switch**, **TextInput**, as well as from third party library components, e.g. **Input, Picker, DateTimePicker**, many more options
- Such components typically have built-in validation props (such as **maxLength** prop for **TextInput**)

Handling forms

- Typically store form values in local component state (controlled form)
- Use a prop on the component such as **onValueChange** for **Picker** to update the state when user updates the value
- Set up a **Button** or other UI component that lets user submit the form, and write code to handle the form submission

Workshop 2 Assignment

In this workshop, you will:

- Add a **modal** to the **CampsiteInfoScreen**
- Add a **form** to the newly created **modal**
- Update the Redux **store** when user submits the **form**

You must have finished all the exercises for this week before you can begin the Workshop Assignment.

You will be split up into groups to work on the assignment together.
Talk through each step out loud with each other, code collaboratively.
If your team spends more than 10 minutes trying to solve one problem, ask your instructor for help!

Check-out

- Submit your assignment files in the learning portal:
- Wrap up – Retrospective
  - What went well
  - What could improve
  - Action items
- If there is time remaining, review any remaining questions from this week, or start on the next week, or work on your Portfolio Project.