Ubiquitous and Mobile Computing
CS 528: React Navigation

William Mosby
Trusting Inekwe

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
Background

- Extension of React Native
- Born from the need to have an easy-to-use navigation solution based on JavaScript
Background

- Easy navigation using Stack implementation (LIFO)
- Animations
- Open source
- Funded by EXPO
- Easy (OTA) updates
- Easy to debug
- Customizable
Specific Problem

- Navigating between screens was an issue for React Native programmers
Typical Use Case

- Navigating between pages in a react-native app
- Passing parameters between pages
- Saving the states of pages
- Redirecting navigation
- Custom navigation animations
Real World Example

- Discontinuity
- Yeti
- Survey
- Golden Owl Consulting
- Maybe
  - Netflix
  - New York Times
  - Khan Academy
How it works

Router API Concept Map

Remember: Routers are often composable, and can delegate to child routers

**URI Events**
URIs that the app is expected to handle, either at wakeup or during app execution

```
router.getActionForPathAndParams(path, params);
```

**App Actions**
Actions that come from user while the app is running

```
router.getStateForAction(action, lastState);
```

**Path and Params**
Relative path and query parameters

```
router.getPathAndParamsForState(state);
```

**URI Output**
Used for sharing links or displaying updated URI in browser bar

**Navigation State**
A list of routes, and an index that points to an active route

**State Logging and Persistence**
Log user navigation behavior and persist the nav state to disk to restore the deep navigation state after refresh
const CourseNavigator = createStackNavigator(
{
Module: Module,
Quiz: QuizScreen,
Video: VideoScreen,
AssetLoading: AssetLoading,
Asset: Asset,
QuizView: QuizView,
},
// $FlowFixMe react-navigation has a bad typedef
{
headerMode: "none",
navigationOptions: {
headerVisible: false
}
}
);

const defaultGetStateForAction = CourseNavigator.router.getStateForAction;

CourseNavigator.router.getStateForAction = (action: NavigationAction, state) => {
console.log("in nav")
console.log(action)
let routeName: ?string = _.get(action, "routeName", null);
if(state &&
(action.type === StackActions.PUSH || action.type === NavigationActions.NAVIGATE) &&
action.routeName === "Module"||

const params = action.params;
console.log("in nav2")
if (params == null) {
  routeName = "Module"
}
else if(params.type === 'quiz'){
  routeName = "Quiz"
}else if(params.type === 'asset'){
  routeName = "AssetLoading"

}
// $FlowFixMe this is perfectly safe but flow handles spreads wonkily
const newAction: NavigationAction = {...action, ...{routeName}}
console.log(newAction);
return defaultGetStateForAction(newAction,state)
<TouchableOpacity
    style={[styles.sheet.buttons]}
    onPress={() => props.navigation.navigate("AboutUs")}
>
    <Text style={styles.sheet.textButtons}>
      <FormattedMessage defaultMessage="About Us" id="About" />
    </Text>
</TouchableOpacity>

<TouchableOpacity
    style={[styles.sheet.buttons]}
    onPress={() => props.navigation.navigate("Help")}
>
    <Text style={styles.sheet.textButtons}>
      <FormattedMessage defaultMessage="Help" id="Help" />
    </Text>
</TouchableOpacity>

<TouchableOpacity
    style={[styles.sheet.buttons]}
    onPress={() => props.navigation.navigate("Preferences")}
>
    <Text style={styles.sheet.textButtons}>
      <FormattedMessage defaultMessage="Preferences" id="preferences" />
    </Text>
</TouchableOpacity>
References

● Retrieved from: 
  https://reactnavigation.org/docs/en/pitch.html

● Retrieved from: 