Navigation in React Native

This module will cover basic navigation patterns like stack, tab, and drawer as well some complex topics like nested navigation.

Common Navigation Patterns

Reusing navigation patterns is a very common practice as it improves user journey. Users of the application don't have to "learn" how to navigate the app. Instead, we may use navigation patters which they already know.

a) Tab Navigation

Uses a tab bar to allow users to switch between screens, and the key functionalities of the application are contained within different tabs. See this (https://reactnavigation.org/docs/bottom-tab-navigator).

b) Stack Navigation

Instead of using tabs, the user has to "go through" every screen. Similar to the stack data-structure, when the user navigates from one screen to the next, the previous screen is pushed to the stack. Consequently, when going back a page, the last screen is popped from the stack and displayed. This navigation pattern is usually employed when the screens are functionality wise related to one another. See this (https://reactnavigation.org/docs/stack-navigator/).

c) Drawer Navigation

This navigation paradigm is similar to the tab navigation, however instead of using a fixed tab bar, it employees a hidden drawer. This can be though of as a pane containing a menu through which users can switch between screens. See this (https://reactnavigation.org/docs/drawer-navigator).

Navigation in Practice

In plain React Native, there is no concept of navigation and only the entrypoint of the application is rendered. Every aspect of navigation needs to be implemented through this entrypoint.

Luckily for us, libraries like react-navigation make life easier. It ships with some common navigators which implement the above mentioned navigation patterns, so that lazy developers like us can only focus on key aspects of development like what should be implemented within the screens, rather than implementing the transitions between the screens themselves.

Consider an example of stack navigation as below. We create 2 components <ScreenOne> and <ScreenTwo> which can be easily rendered in a stack fashion by passing them as children to the <Stack.Navigator> in the <NavigationContainer> of the <App> component. Note that <Stack.Navigator> and <Stack.Screen> are initialized by calling the createStackNavigator() function.

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

const ScreenOne = () => (
  <View style={styles.layout}>
    <Text style={styles.title}>First screen</Text>
  </View>
);

const ScreenTwo = () => (
  <View style={styles.layout}>
    <Text style={styles.title}>Second screen</Text>
  </View>
);

const Stack = createStackNavigator();

const App = () => (
  <NavigationContainer>
    <Stack.Navigator>
      <Stack.Screen name="FirstScreen" component={ScreenOne} />
      <Stack.Screen name="SecondScreen" component={ScreenTwo} />
    </Stack.Navigator>
  </NavigationContainer>
);

export default App;

const styles = StyleSheet.create({
  layout: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  title: {
    fontSize: 32,
    marginBottom: 16,
  },
});
```

Similar examples for other navigation patterns may be seen from the documentation of [React Navigator (https://reactnavigation.org/docs/getting-started)](https://reactnavigation.org/docs/getting-started).