Navigation

Bala Krishna Ragala

The Problem

- React Native doesn't have a built-in API for navigation like a web browser does. React Navigation provides this for you, along with the iOS and Android gestures and animations to transition between screens.

React Navigation

- React Navigation provides an easy to use navigation solution, with the ability to present common **stack navigation and tabbed navigation patterns on both iOS and Android**.
- A complete JavaScript implementation
- Provides the greatest amount of configurability as well as flexibility when integrating with state management libraries such as redux.

Alternative libraries

- **react-native-router-flux**: this library is based on React Navigation but provides you with a different API to interact with it.
- **react-native-navigation**: uses the underlying native APIs on iOS and Android, this is a popular alternative to React Navigation and worth considering if you value adhering to the platform conventions exactly and do not care as much about customization.
- **react-router-native**: this library is somewhat incomplete, but if you're familiar with the React Router API already and have simple requirements for your app, this might work for you.

Why React Navigation

- Everything is written in JavaScript on top of React Native primitives
- JavaScript has a lot of benefits:
  - Easy OTA updates
  - Debuggable
  - Customizable
- Most apps heavily customize navigation, to do this with an API that wraps native navigation you will need to write a lot of native code.
- It's easy to write your own navigators that integrate cleanly with standard navigators, or to fork the standard navigators and create your own version of them with the exact look and feel you want in your app.

https://reactnavigation.org/docs/en/pitch.html

Setup

npm install react-navigation

Or

yarn add react-navigation

StackNavigator

- Provides a way for your app to transition between screens where each new screen is placed on top of a stack.
- By default the StackNavigator is configured to have the familiar iOS and Android look & feel:
  - new screens slide in from the right on iOS
  - fade in from the bottom on Android
- Signature

```
createStackNavigator(RouteConfigs, StackNavigatorConfig);
```

Usage

```
// In App.js in a new project

import React from 'react';
import { View, Text } from 'react-native';
import { createStackNavigator } from 'react-navigation';

class HomeScreen extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
        <Text>Home Screen</Text>
      </View>
    );
  }
}

export default createStackNavigator({
  Home: {
    screen: HomeScreen
  },
});
```

StackNavigator - RouteConfigs

- The route configs object is a mapping from route name to a route config, which tells the navigator what to present for that route

StackNavigatorConfig

- Options for the router:
  - **initialRouteName** - Sets the default screen of the stack. Must match one of the keys in route configs.
  - **initialRouteParams** - The params for the initial route
  - **navigationOptions** - Default navigation options to use for screens
  - **paths** - A mapping of overrides for the paths set in the route configs
- Visual options:
  - **card** - Use the standard iOS and Android screen transitions. This is the default.
  - **modal** - Make the screens slide in from the bottom which is a common iOS pattern. Only works on iOS, **has no effect on Android**.
- Many more config (https://reactnavigation.org/docs/en/stack-navigator.html)

navigationOptions

- **title** - String that can be used as a fallback for **headerTitle**
- **headerTitle -** String, React Element or React Component used by the header
- **headerBackImage** - React Element or Component to display custom image in header's back button. When a component is used, it receives a number of props when rendered (tintColor, title). Defaults to Image component with react-navigation/views/assets/back-icon.png back image source, which is the default back icon image for the platform (a chevron on iOS and an arrow on Android).
- **headerBackTitle** - Title string used by the back button on iOS, or null to disable label. Defaults to the previous scene's headerTitle.

navigationOptions [contd...]

- **headerRight** - React Element to display on the right side of the header.
- **headerLeft -** React Element or Component to display on the left side of the header.
- **headerStyle** - Style object for the header
- **headerTitleStyle** - Style object for the title component
- **headerBackTitleStyle** - Style object for the back title
- **headerTintColor** - Tint color for the header

Configuring the header bar

- A screen component can have a **static** property called **navigationOptions** which is either an **object or a function** that returns an object that contains various configuration options.
- StackNavigator uses platform conventions by default, **so on iOS the title will be centered and on Android it will be left-aligned**.

Adjusting header styles

- three key properties to use when customizing the style of your header:
  - headerStyle,
  - headerTintColor, and
  - headerTitleStyle
- **headerStyle**: a style object that will be applied to the View that wraps the header. If you set backgroundColor on it, that will be the color of your header
- **headerTintColor**: the back button and title both use this property as their color.

Sharing common navigationOptions across screens

- You can define the navigationOptions at the navigator definition level, which will be shared with every screen
- You can override these shared settings at screen level by re-defining the same properties

Replacing the title with a custom component

- Use **headerTitle** property of **navigationOptions** to assign a custom component to override default title component from react navigation

Moving between screens

- **navigation** prop is passed in to every screen component (definition) in StackNavigator
  - navigate - go to another screen, figures out the action it needs to take to do it
  - goBack - close active screen and move back in the stack
  - addListener - subscribe to updates to navigation lifecycle
  - isFocused - function that returns true if the screen is focused and false otherwise.
  - state - current state/routes
  - setParams - make changes to route's params
  - getParam - get a specific param with fallback
  - dispatch - send an action to router
- **Note: navigation prop is not passed in to all components; only screen components receive this prop automatically**

Passing parameters to routes

- 2 ways
- **First Way** - Pass params to a route by putting them in an object as a second parameter to the navigation.navigate function: this.props.navigation.navigate('RouteName', **{ /* params go here */ }**)
- **Second Way** - Read the params in your screen component: **this.props.navigation.state.params**.

Tab navigation

- most common style of navigation in mobile apps
- can be tabs on the bottom of the screen or on the top below the header

Example

```
import React from 'react';
import { Text, View } from 'react-native';
import { createBottomTabNavigator } from 'react-navigation';

class HomeScreen extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Home!</Text>
      </View>
    );
  }
}

class SettingsScreen extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Settings!</Text>
      </View>
    );
  }
}

export default createBottomTabNavigator({
  Home: HomeScreen,
  Settings: SettingsScreen,
});
```

Tab Navigator Config

- https://reactnavigation.org/docs/en/tab-navigator.html

Question Time

- Different types of navigators?
- How do you navigate to different screen?
- Which method creates StackNavigator?
- How do you pass params to navigating component?
- How do you read params from navigated component?
- When there are multiple screens how do you set the initial screen?
- How do you overwrite the shared navigation options?
- Can we use StackNavigator and TabNavigator together?