**Exploring Tab Navigation**

You learned how to use the Tab Navigator from React Navigation in earlier videos. You also built the tab navigator within the Little Lemon app to move between the screens. In this reading, you will explore the code used to build the Tab Navigator in detail.

Tab Navigation is a standard navigation methodology in mobile applications. React Navigation provides several UI options for Tab Navigation.

In this example, you will use React Navigation’s bottom tabs library for tab navigation.

**Installation**

To begin using the tab navigator, you must first install the following package within your application.

1

npm install @react-navigation/bottom-tabs

This command will install the **bottom-tabs** package from React Navigation.

**Imports**

Within your code, the first step is to ensure that you import the **createBottomTabNavigator** as shown below:

1

import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

All the other necessary imports are shown below:

1

2

3

4

5

import { NavigationContainer } from '@react-navigation/native';

import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

import MenuScreen from './Screens/MenuScreen';

import WelcomeScreen from './Screens/WelcomeScreen';

You will continue to use the same **NavigationContainer** to wrap around the entire app, just like you did when setting up the Stack Navigator. In addition, the screens that are part of the tab navigation are also imported.

In this case, it is the **MenuScreen** and the **WelcomeScreen**.

**Configuring Tab Navigation**

The next step is to create an instance of the **createBottomTabNavigator** as shown below:

1

const Tab = createBottomTabNavigator();

This will return an object with two properties; the **navigator** and the **screen**. Again this is very similar to the Stack Navigator API you have already learned about.

Within your root **App** component, you then add the following code:

1

2

3

4

5

6

7

8

9

10

11

export default function App() {

  return (

    <NavigationContainer>

      <Tab.Navigator>

        <Tab.Screen name="Welcome" component={WelcomeScreen} />

        <Tab.Screen name="Menu" component={MenuScreen} />

      </Tab.Navigator>

    </NavigationContainer>

  );

}

That’s it! The tab navigator is set up. You have just added two screens for navigation; the **Welcome** screen and the **Menu** screen. They are wrapped within the **Tab.Navigator** component, and the overall app is wrapped within the **NavigationContainer***.*

**Customizing Tab Navigation**

To customize your tab navigator further, you can provide icons for each tab, colors for the icons, and change the colors based on whether the icon is active or inactive.

To add icons, use the **vector-icons** package that comes as default with Expo. There is no additional setup needed to utilize it. You can use any library of your choice, but for this demonstration, use an open-source library called **Ionicons**.

To do this, you will first import **Ionicons** from the **vector-icons** package:

1

import { Ionicons } from '@expo/vector-icons';

**Screen Options**

Next, within the **Navigator***,* you can add the customization using the **screenOptions** prop, as shown below:

17

18

19

20

21

22

23

24

25

26

27

15

16

12

13

14

10

11

7

8

9

5

6

3

4

1

2

        },

        tabBarActiveTintColor: 'tomato',

        tabBarInactiveTintColor: 'gray',

      })}>

     <Tab.Screen name="Welcome" component={WelcomeScreen} />

      <Tab.Screen name="Menu" component={MenuScreen} />

    </Tab.Navigator>

    </NavigationContainer>

  );

}

          }

          return <Ionicons name={iconName} size={size} color={color} />;

              : 'ios-information-circle-outline';

          } else if (route.name === 'Menu') {

            iconName =  'ios-list';

            iconName = focused

              ? 'ios-information-circle'

          let iconName;

          if (route.name === 'Welcome') {

        screenOptions={({ route }) => ({

        tabBarIcon: ({ focused, color, size }) => {

    <NavigationContainer>

      <Tab.Navigator

export default function App() {

  return (

Notice here that you can provide the **tabBarIcon**, **tabBarActiveTintColor** and **tabBarInactiveTintColor** values to the **screenOptions***.*

This reading taught you how to configure the tab navigator and customize it based on the user’s needs.

Mark as completed

Like

Dislike

Report an issue