06016323 Mobile Device Programming

CHAPTER 6 : NAVIGATION (PART 2)

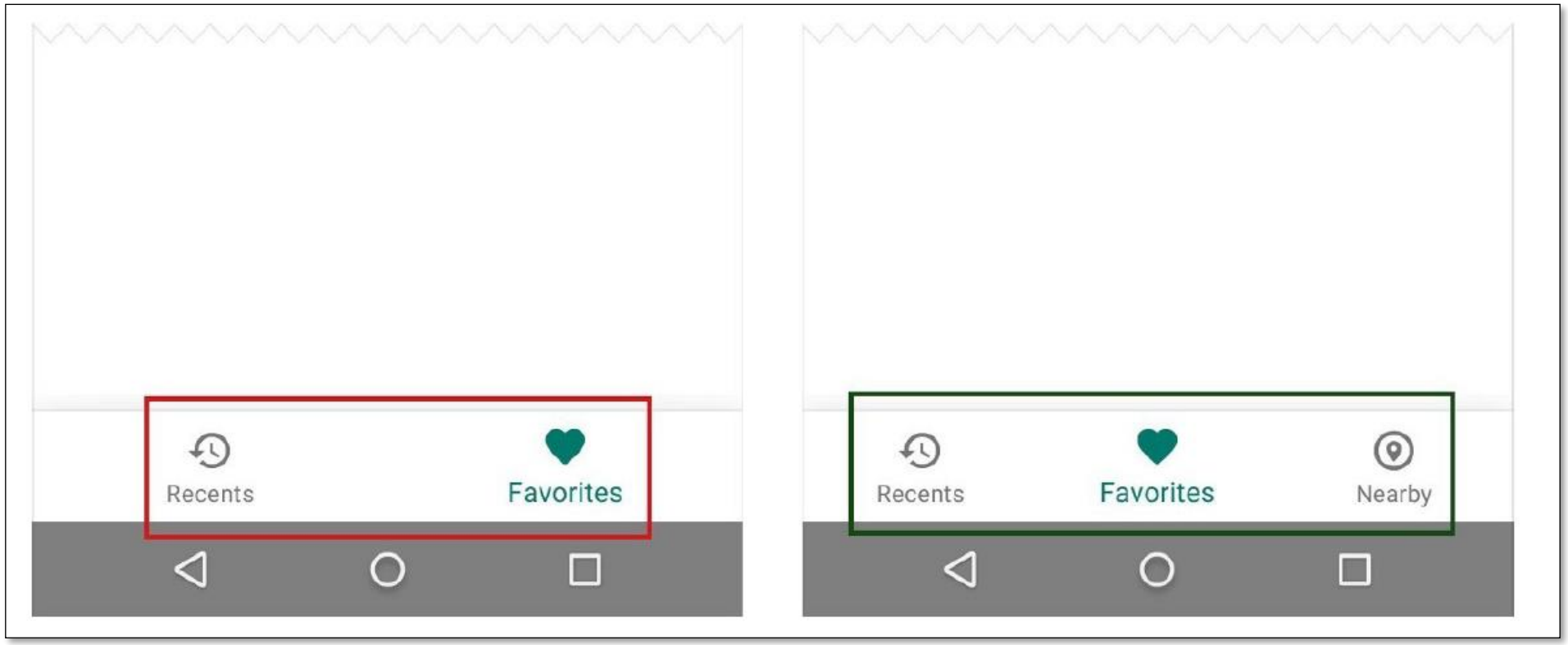
React Navigation

- Stack Navigation
- Tab Navigation
- Drawer Navigation

Tab Navigation

- การใช้ Stack navigation ในการเปลี่ยนหน้าจอเพียงอย่างเดียว อาจไม่รองรับการทำงานของโปรแกรมที่ต้องการได้
- Tab navigation อนุญาตให้ผู้ใช้สามารถเปลี่ยนหน้าจอในระดับ root ได้
- เหมาะกับกรณีที่มีหน้าจอหลายๆ หน้าจอที่มีความสำคัญเท่าๆ กัน
- Tab bar สามารถแสดงที่ส่วนด้านบนและด้านล่างของจอได้

Tab Navigation

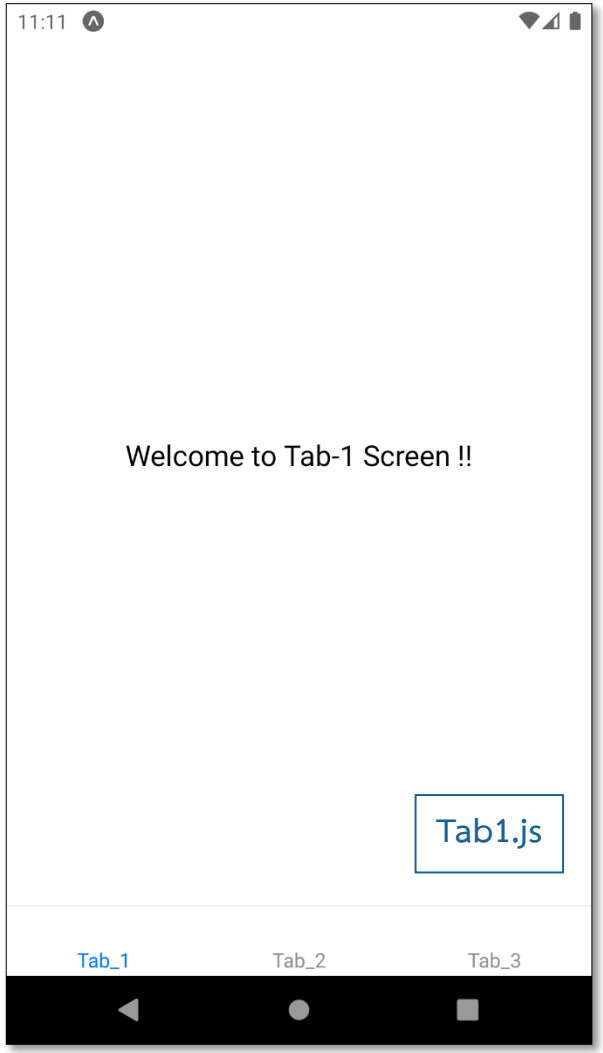


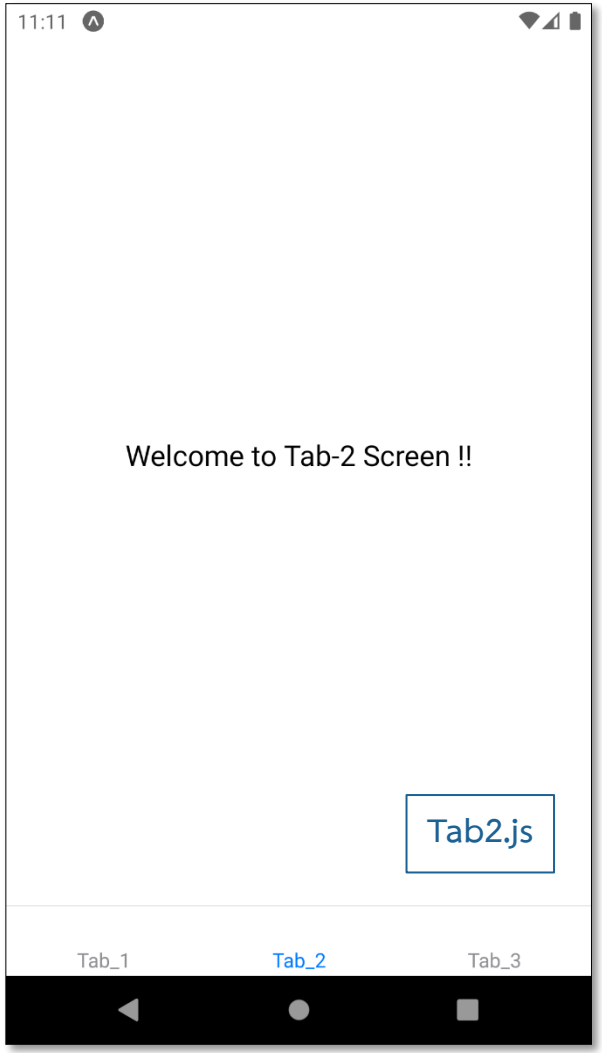
ติดตั้ง react-navigation-tabs

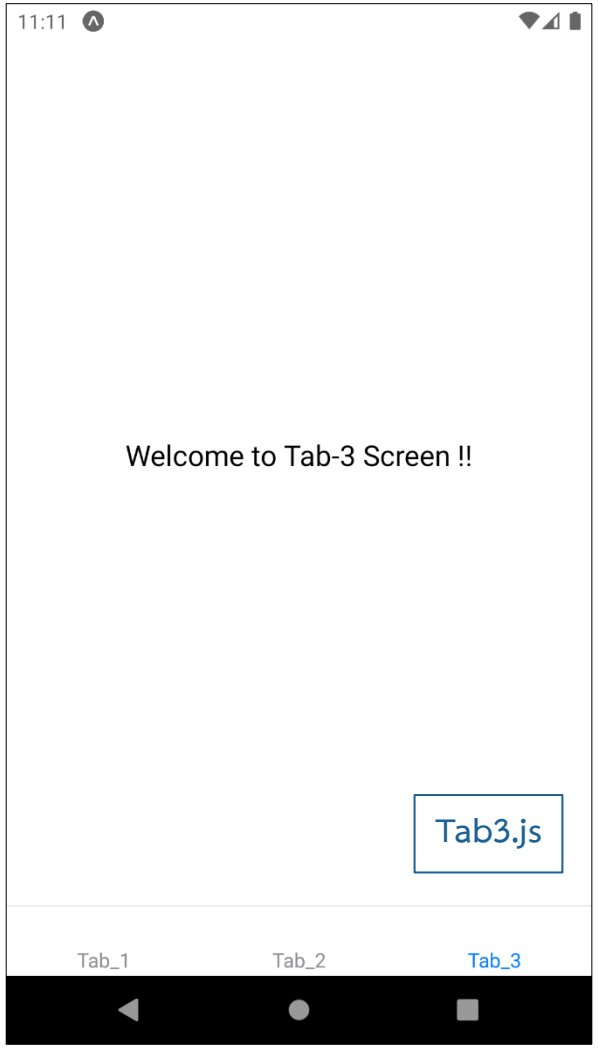
- สำหรับ react-navigation เวอร์ชัน 6 ขึ้นไป ต้องมีการติดตั้งไลบรารีเพิ่มเติม เพื่อทำการสร้าง Navigator ในรูปแบบต่างๆ (ในที่นี้ จะสร้าง Tab Navigator)
  - npm install --save @react-navigation/bottom-tabs หรือ
  - expo install @react-navigation/bottom-tabs
- ศึกษา React navigation เพิ่มเติมได้ที่ :

https://reactnavigation.org/docs/getting-started

ตัวอย่างแอปพลิเคชัน







ตัวอย่างโปรแกรม Tab1.js

```
import React from "react";
import { View, Text, StyleSheet } from "react-native";

const Tab1 = (props) => {
  return (
    <View>
      <Text>Welcome to Tab-1 Screen !!</Text>
    </View>
  );
};
export default Tab1;
```

การสร้าง Tab Navigator

- หากต้องการสร้าง Tab navigator ทำได้โดยเรียก (คล้ายกับ Stack navigator)
  - createBottomTabNavigator (**)
  - createMaterialBottomTabNavigator
  - createMaterialTopTabNavigator
- เริ่มต้น ต้องทำการ import สิ่งที่ใช้ในการทำ tab navigation ที่ต้องการ เช่น
  - import { createBottomTabNavigator } from "@react-navigation/bottom-tabs"; // v.6
  - import { NavigationContainer } from "@react-navigation/native"
- สร้าง tab navigator ด้วย createBottomTabNavigator()
- เมื่อจะนำ Navigator ไปใช้ ต้องอยู่ภายใต้ NavigationContainer

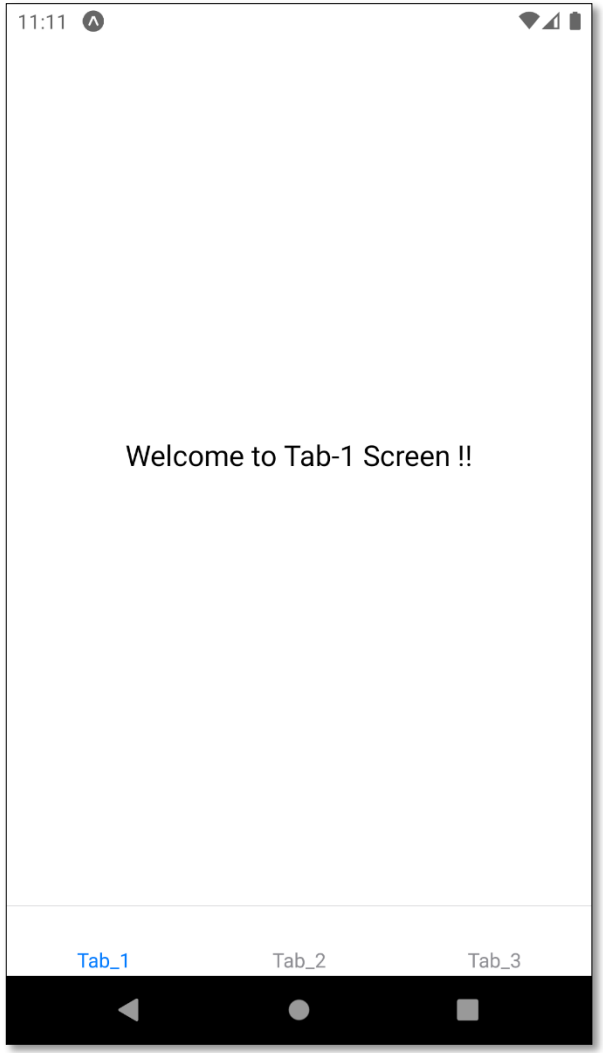
createBottomTabNavigator

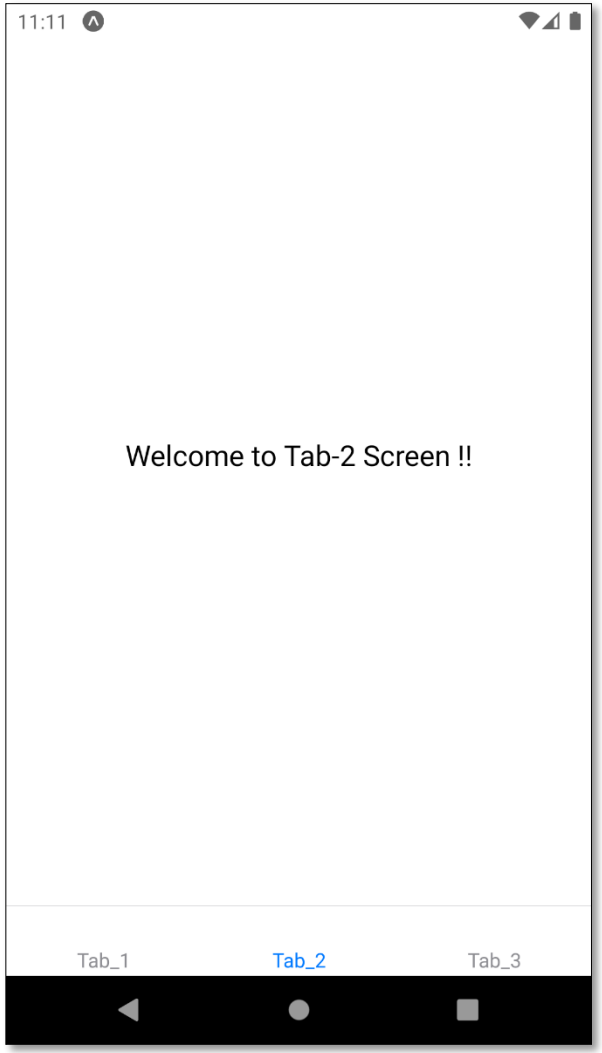
- เป็นฟังก์ชันที่ใช้สร้าง Bottom tab navigator ซึ่งจะคืนค่าอ็อบเจคต์ที่มี property 2 อย่าง (เหมือน Stack navigator)
  - Screen เป็นคอมโพเนนต์ที่ใช้ในการกำหนดเส้นทางสำหรับการทำ navigation โดยมี prop ที่สำคัญ เช่น
    - name : เป็นชื่อของเส้นทางที่ Navigator ใช้อ้างถึง
    - component : อ้างถึงคอมโพเนนต์ที่ต้องการเรนเดอร์ (หน้าจอที่จะแสดงผล)
    - options : ใช้กำหนดรายละเอียดเกี่ยวกับการแสดงผลของหน้าจอนั้นๆ
  - Navigator จะประกอบด้วย Screen อยู่ภายใน ใช้กำหนดค่าสำหรับควบคุมการทำ navigation มี prop ที่สำคัญ เช่น
    - initialRouteName : ใช้กำหนดเส้นทางเริ่มต้นของการทำ Navigation
    - screenOptions : ใช้กำหนดรายละเอียดการแสดงผลโดยรวมของ Screen ที่ Navigator ดูแลทั้งหมด

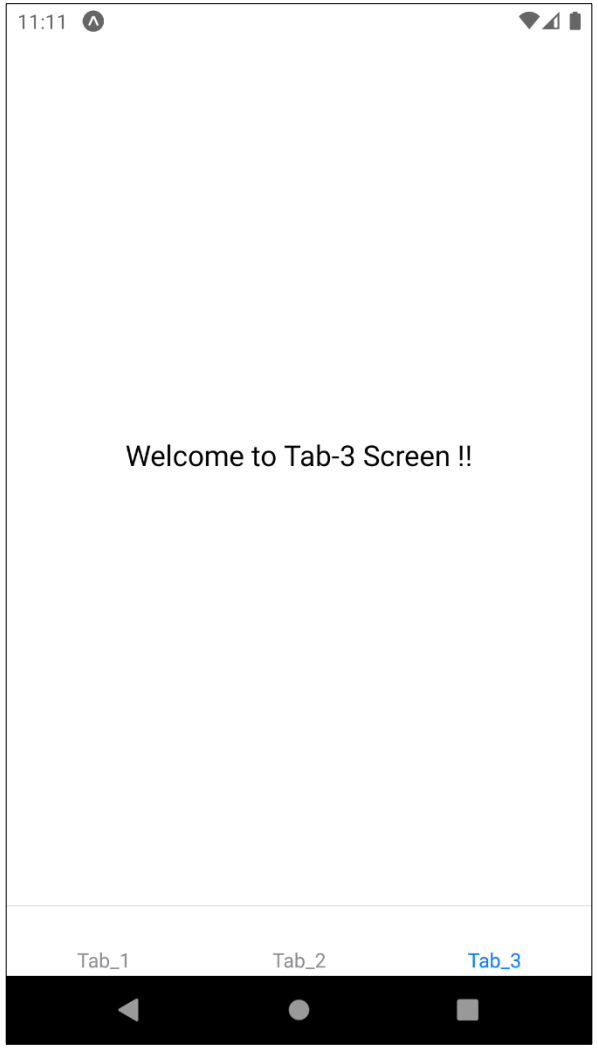
ตัวอย่างโปรแกรม App.js

```
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { NavigationContainer } from "@react-navigation/native";

import Tab1 from "../screens/Tab1";
import Tab2 from "../screens/Tab2";
import Tab3 from "../screens/Tab3";

const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator initialRouteName="T1">
        <Tab.Screen name= "Tab_1" component={Tab1} />
        <Tab.Screen name= "Tab_2" component={Tab2} />
        <Tab.Screen name= "Tab_3" component={Tab3} />
      </Tab.Navigator>
    </NavigationContainer>
  ); }
```

ตัวอย่างแอปพลิเคชัน





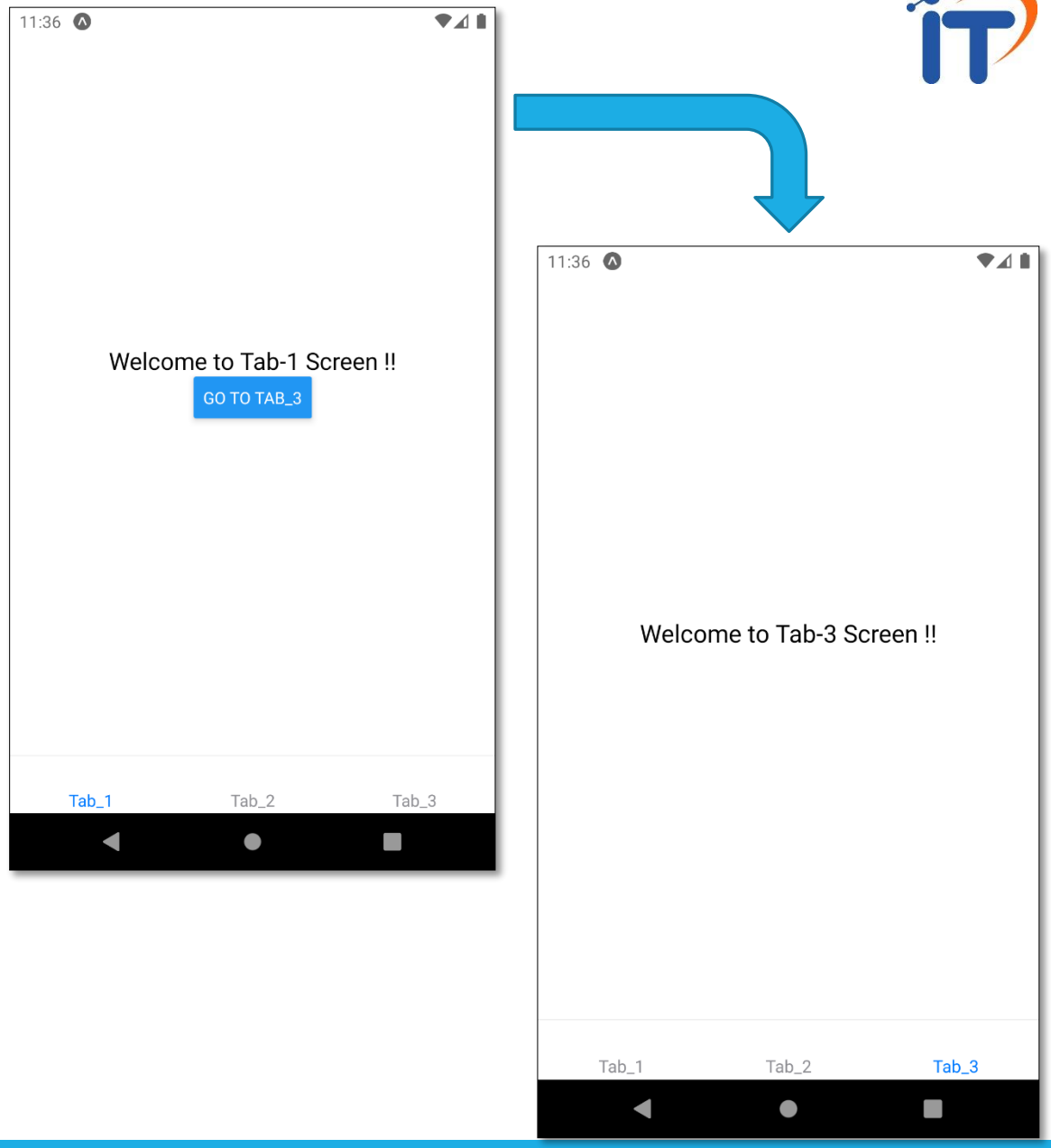


การ Navigate ระหว่าง tab

- เมื่อกดที่ปุ่ม tab ด้านล่าง ก็จะทำการเปลี่ยนหน้าจอตามที่กำหนด
- นอกจากนี้ ยังสามารถเปลี่ยน tab ได้โดยการเรียก navigate ผ่าน navigation props ได้ (คล้ายกับ stack navigation)
- เมธอดพื้นฐานของ tab navigation
  - navigate
  - goBack

ตัวอย่างโปรแกรม Tab1.js

```
import React from "react";
import { View, Text, StyleSheet, Button } from "react-native";

const Tab1 = ({navigation) => {
  return (
    <View>
      <Text>Welcome to Tab-1 Screen !!</Text>
      <Button
        title="Go to Tab_3"
        onPress={ () => { navigation.navigate("Tab_3"); } }
      />
    </View>
  );
};
export default Tab1;
```

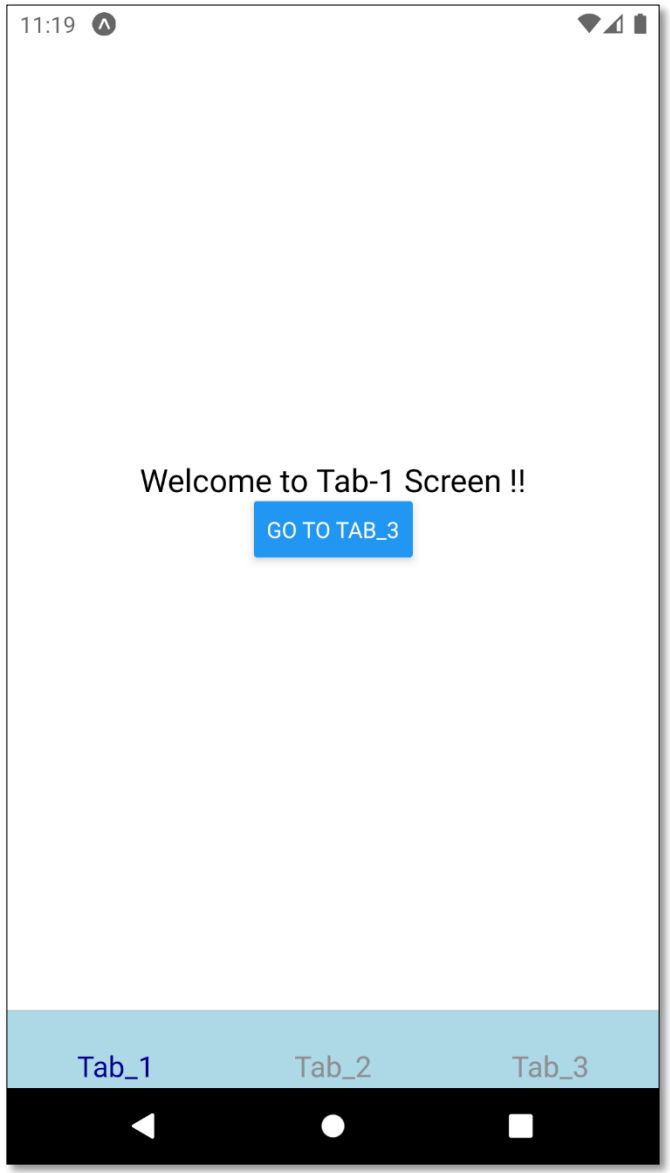


Options prop : ปรับแต่งค่าการแสดงผล

- กรณีที่ต้องการปรับแต่งค่าโดยรวมของ navigation สามารถทำได้คล้ายกับ stack navigation
  - สามารถกำหนดค่า options ได้ในคอมโพเนนต์ Screen
  - ตั้งค่าผ่าน options ที่หลากหลายได้ (ดูเพิ่มเติมใน React Navigation Docs)
  - ตัวอย่าง key properties ที่ใช้การกำหนดค่าการแสดงผลของ Bottom tab screen
    - title
    - tabBarIcon
    - tabBarStyle
    - tabBarActiveTintColor
  - กรณีที่ต้องการใช้ค่า options ร่วมกันในทุก Screen สามารถปรับแต่งค่าได้ผ่าน screenOptions ในคอมโพเนนต์ Navigator

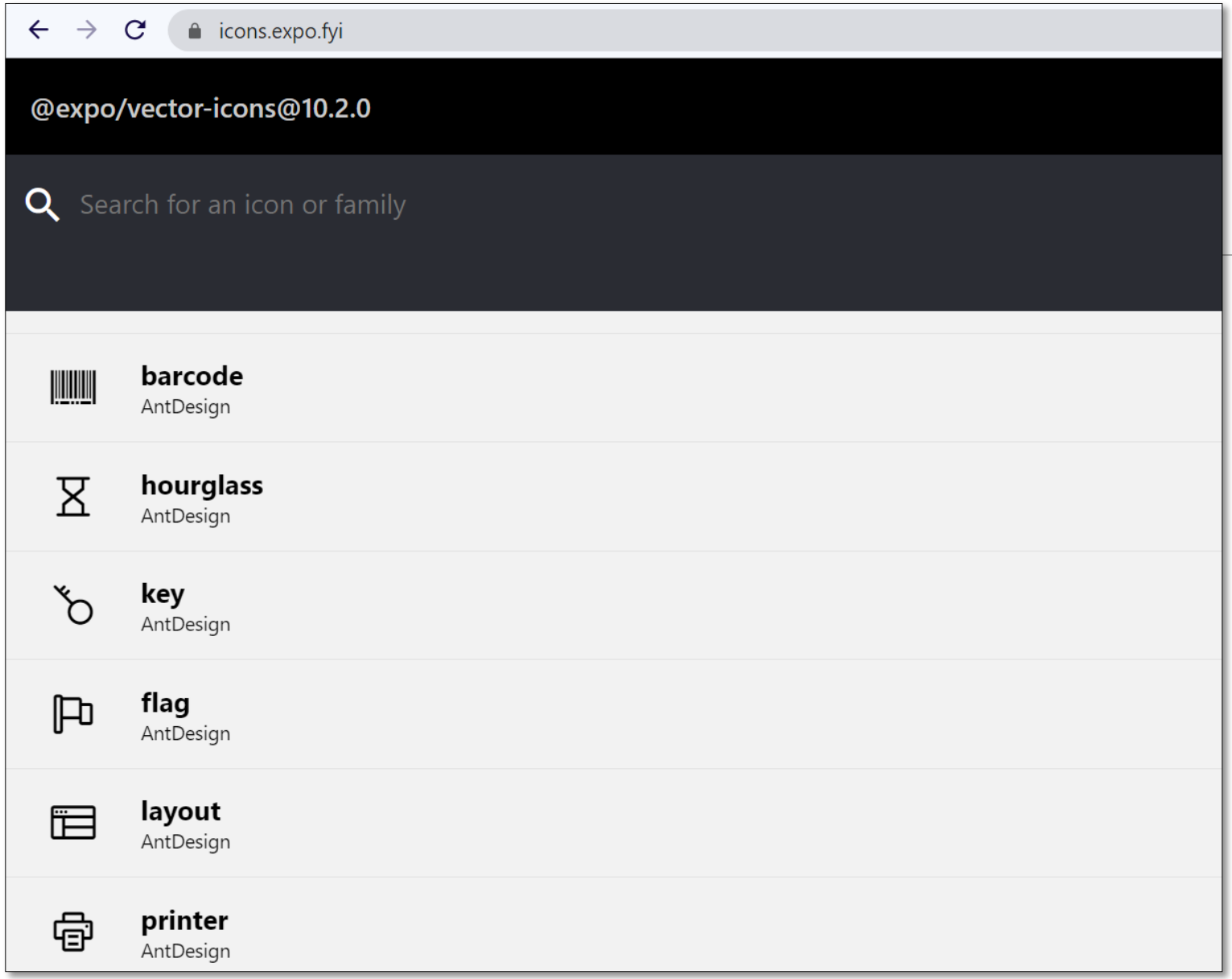
ตัวอย่างโปรแกรม App.js

```
// import ...
const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator
        screenOptions={{
          tabBarActiveTintColor: "darkblue",
          tabBarStyle: { backgroundColor: "lightblue" },
          tabBarLabelStyle: { fontSize: 15 },}} >
        <Tab.Screen name="Tab_1“ component={Tab1} />
        <Tab.Screen name="Tab_2“ component={Tab2} />
        <Tab.Screen name="Tab_3“ component={Tab3} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```



tabBarIcon

- เราสามารถกำหนดไอคอนที่ tab ผ่าน tabBarIcon ใน options prop ได้
- สามารถใช้ @expo/vector-icons ได้ (ต้องติดตั้ง @expo/vector-icons)
  - import { *icon-family* } from "@expo/vector-icons";
  - ตัวอย่าง *icon-family* เช่น Ionicons, AntDesign เป็นต้น

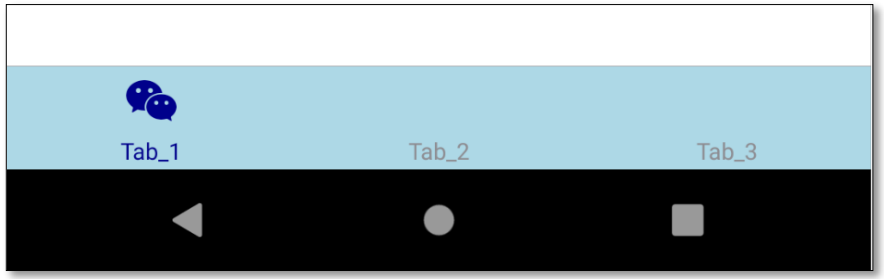


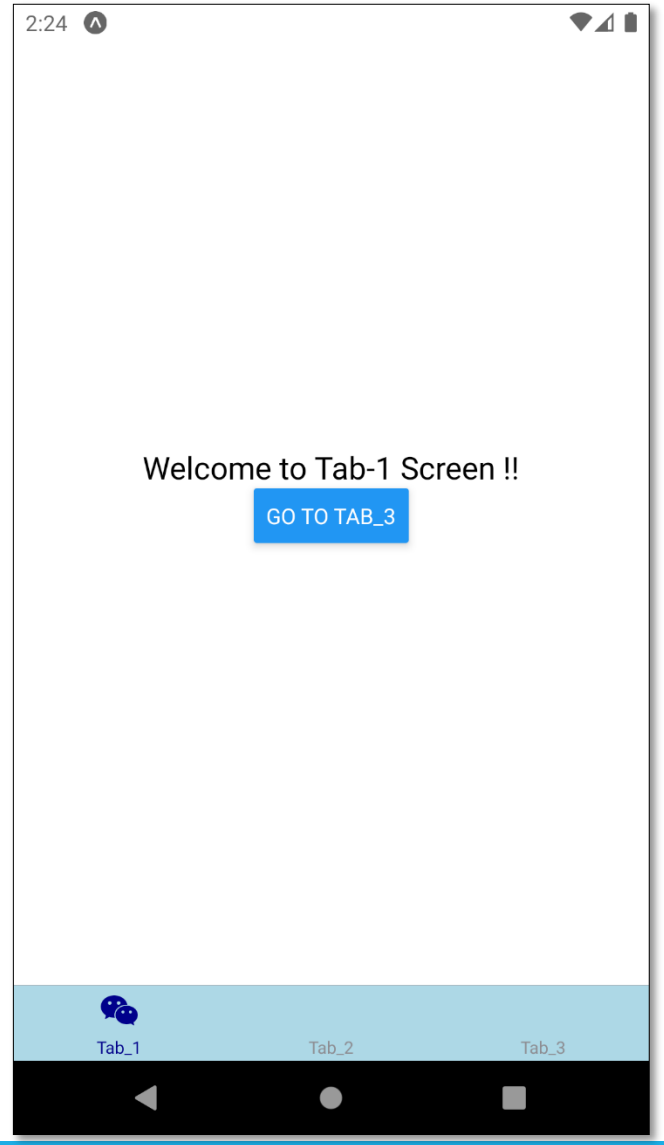
@expo/vector-icons

(https://icons.expo.fyi/)

ตัวอย่างโปรแกรม App.js

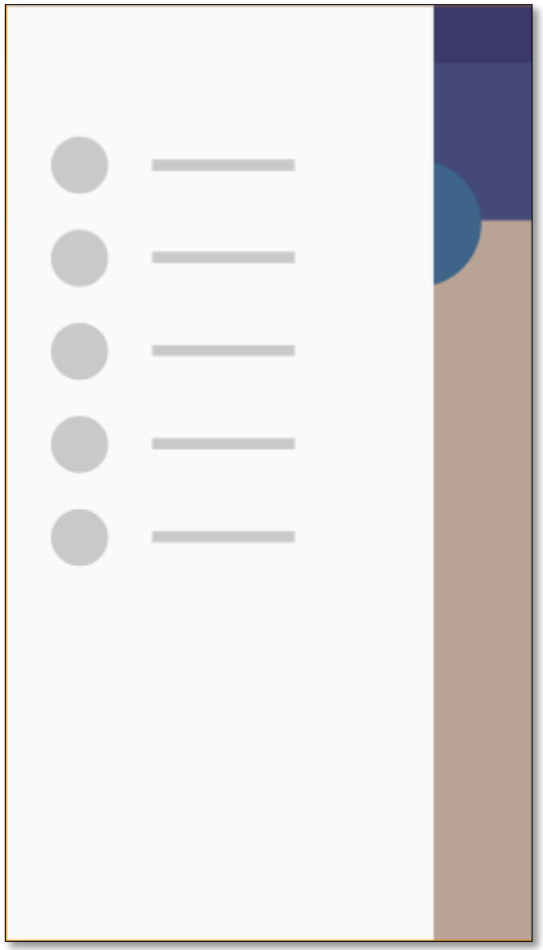
```
// import ...
import { AntDesign } from "@expo/vector-icons";
const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator screenOptions={{ .. }} >
        <Tab.Screen name="Tab_1“ component={Tab1}
              options={{
                  tabBarIcon: ({ color, size }) => {
                       return <AntDesign name="wechat" size={size} color={color} />;
                  },
               }} />
        <Tab.Screen name="Tab_2“ component={Tab2} />
        <Tab.Screen name="Tab_3“ component={Tab3} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```





Drawer Navigation

- เป็นการทำ navigation อีกรูปแบบหนึ่ง ที่มีลักษณะเป็น Slide bar จากด้านซ้ายของจอ
- expo install @react-navigation/drawer
- ติดตั้ง dependencies เพิ่มเติม
  - expo install react-native-gesture-handler react-native-reanimated
- import { createDrawerNavigator } from "@react-navigation/drawer";
- ใช้ createDrawerNavigator() ในการสร้าง Drawer Navigator และสามารถปรับแต่งค่าการแสดงผลได้ ผ่านการกำหนด options prop หรือ screenOptions (คล้ายกับ Stack และ Bottom tab navigator)

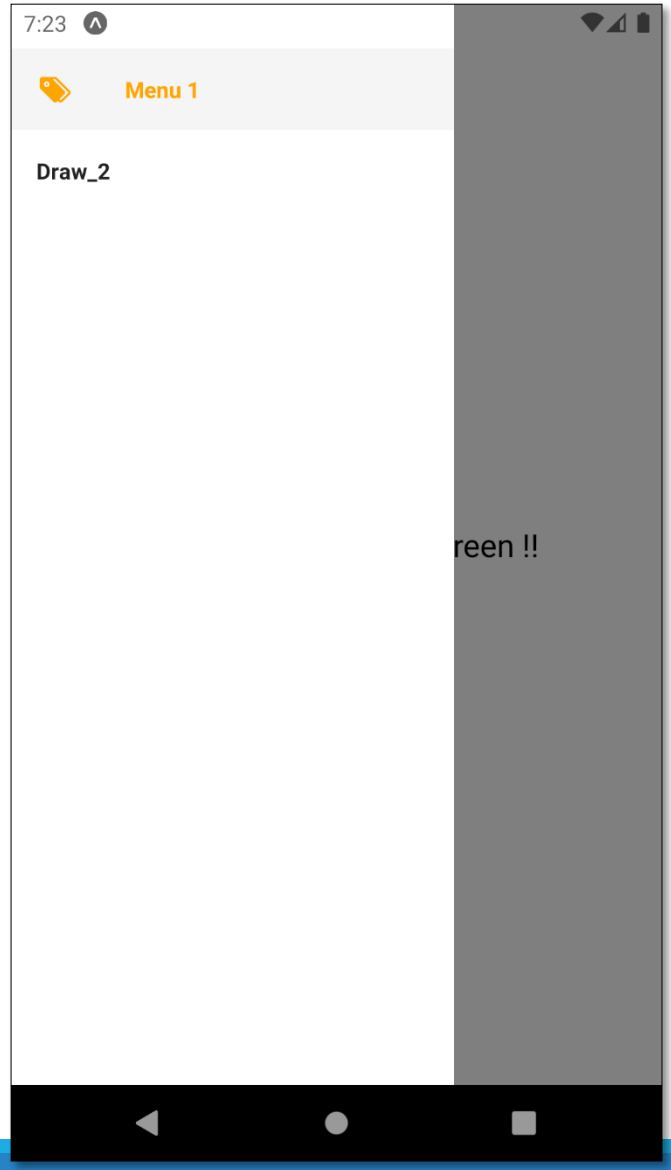


เมธอดพื้นฐานการทำ drawer navigation

- closeDrawer
- goBack
- navigate
- openDrawer
- toggleDrawer

ตัวอย่างโปรแกรมสร้าง Drawer Navigator

```
const Drawer = createDrawerNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Drawer.Navigator
        screenOptions={{ drawerActiveTintColor: "orange", drawerInactiveTintColor: "gray", }} >
        <Drawer.Screen name="Draw_1“ component={Draw1}
          options={{
            drawerLabel: "Menu 1",
            drawerIcon: ({ color }) => { return <AntDesign name="tags" size={24} color={color} />; },
          }} />
        <Drawer.Screen name="Draw_2“ component={Draw2} />
      </Drawer.Navigator>
    </NavigationContainer>
  );
}
```



Nesting Navigation

- เมื่อโปรแกรม/แอปพลิเคชันมีความซับซ้อน โดยมีหน้าจอหลายส่วนและจำเป็นต้องใช้รูปแบบ navigation หลายรูปแบบ
- ในกรณีนี้ เราสามารถใช้ Nesting navigation ได้ โดยทำการซ้อน navigator หนึ่ง อยู่ใน navigator อื่นได้
  - เช่น การซ้อน tab navigator ใน stack navigator เป็นต้น
- รูปแบบการเขียนโปรแกรม เราสามารถสร้างและกำหนดรายละเอียดของ Navigator ผ่านฟังก์ชัน และเมื่อมีการเรียก navigator ซ้อน ก็ทำการกำหนด Component ให้เป็นฟังก์ชันที่สร้างไว้ข้างต้น

Nesting Navigation

- Navigator แต่ละตัว จะมีลำดับการทำ navigation ของตัวเอง
  - เช่น stack ซ้อน stack หากหน้าจอเป็นของ stack navigator ภายใน หากกด Back โปรแกรมจะแสดงหน้าจอก่อนหน้าของ stack นั้น ไม่ยุ่งเกี่ยวกับ stack ภายนอก
- Navigator แต่ละตัวจะมี options เป็นของตัวเอง
  - เช่น title ที่กำหนดใน navigator ภายใน จะไม่กระทบต่อ title ของ navigator ภายนอก
- แต่ละหน้าจอของ navigator หนึ่งๆ จะมี params เป็นของตัวเอง
  - เช่น params ที่ส่งให้ screen ใน navigator ภายใน จะไม่สามารถเข้าถึงได้จาก navigator ภายนอก

Nesting Navigation

- Navigation action จะถูกจัดการด้วย navigator ล่าสุด ถ้าไม่สามารถจัดการได้ navigator ภายนอกจะพยายามจัดการเอง
- Navigator ที่อยู่ภายใน จะสามารถใช้เมธอดเฉพาะของ navigator ภายนอกได้
  - เช่น tab ซ้อนใน stack navigator แล้ว หน้าจอของ tab navigator นั้นจะมีเมธอด push และ replace ใน navigation prop ด้วย
- Navigator ที่อยู่ภายใน จะไม่สามารถรับ event จาก navigator ภายนอกได้
  - เช่น stack ซ้อนใน tab หน้าจอใน stack จะไม่รับ event จาก tab navigator เช่น tabPress
- UI ของ navigator ภายนอกจะถูกเรนเดอร์บน navigator ภายในอีกที

Nesting Navigation

- รูปแบบการเขียนโปรแกรม เราสามารถสร้างและกำหนดรายละเอียดของ Navigator ผ่านฟังก์ชัน และเมื่อมีการเรียก navigator ซ้อน ก็ทำการกำหนด Component ให้เป็นฟังก์ชันที่สร้างไว้ข้างต้น
- ก่อนหน้านี้ เราสร้าง Navigator ที่ App.js แต่เมื่อมีการเรียกใช้ navigator แบบซ้อนกัน เราอาจสร้างไฟล์ในการกำหนด navigator โดยเฉพาะ แล้วให้ App.js มีการเรียกใช้คอมโพเนนต์ของไฟล์ดังกล่าว

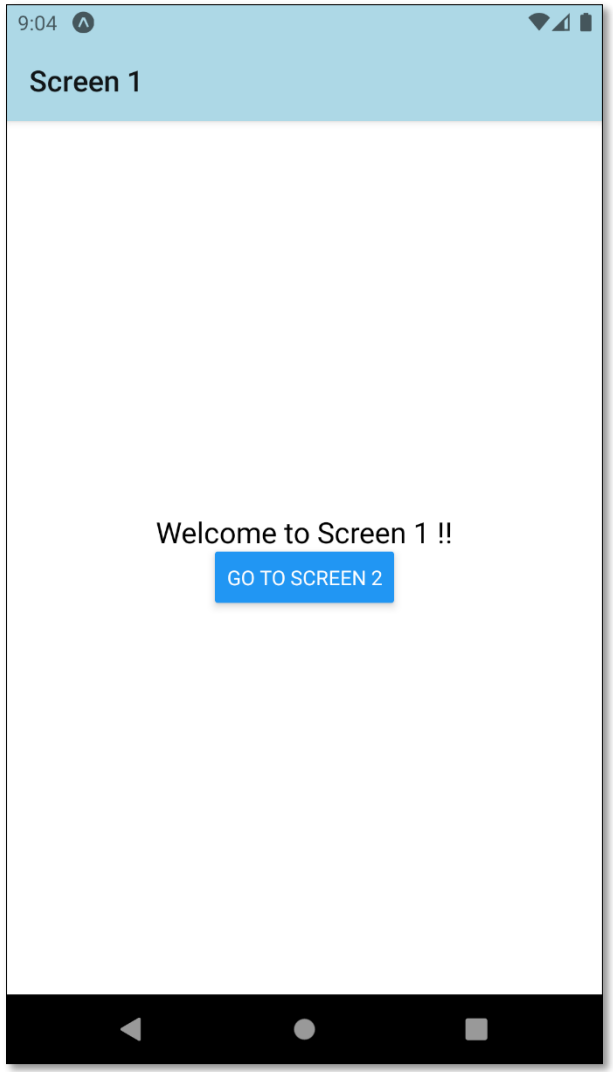
ตัวอย่าง Nesting navigators (MyNavigator.js)

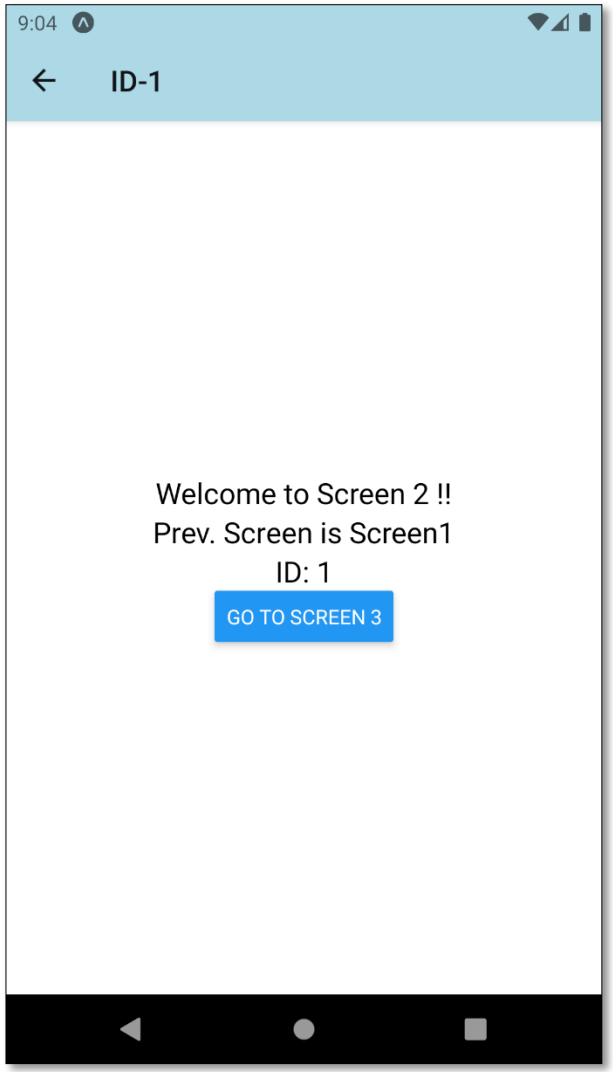
```
// MyNavigator.js
const Stack = createNativeStackNavigator();
const Tab = createBottomTabNavigator();

function MyTab() {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Tab_1" component={Tab1} />
      <Tab.Screen name="Tab_2" component={Tab2} />
      <Tab.Screen name="Tab_3" component={Tab3} />
    </Tab.Navigator>
  ); }
```

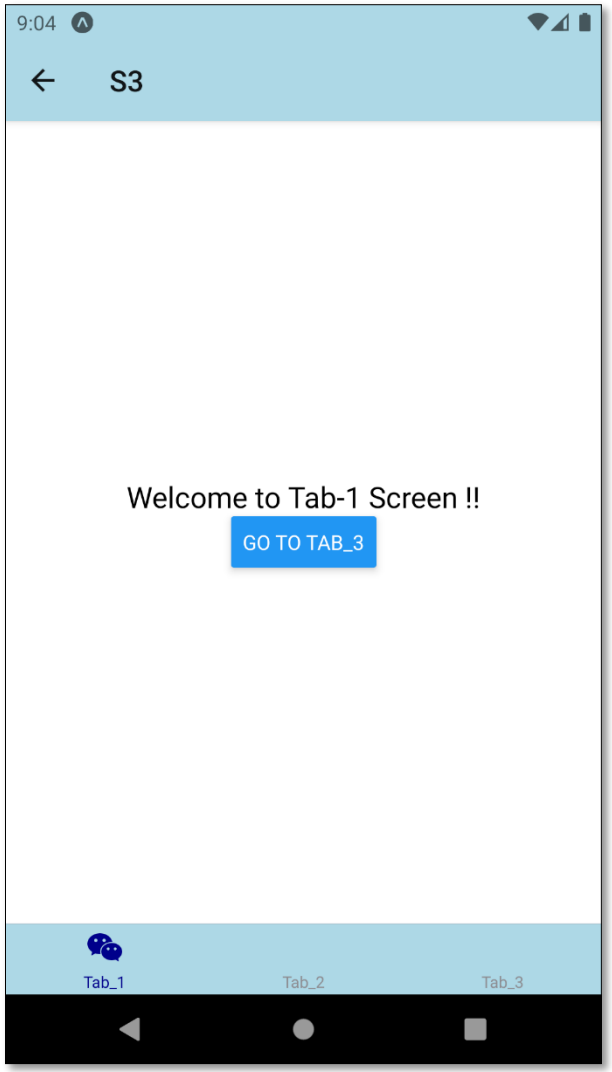
```
//App.js
export default function App() {
  return <MyNavigator />;
}
```

```
// MyNavigator.js (ต่อ)
export default function MyNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator
        screenOptions={{ headerStyle: { backgroundColor: "lightblue" } }} >
        <Stack.Screen name="S1“ component={Screen1} />
        <Stack.Screen
          name="S2“ component={Screen2}
          options={({ route }) => ({
            title: route.params.prev + " ID-" + route.params.id.toString(),})} />
        <Stack.Screen name="S3“ component={MyTab} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

ตัวอย่าง Nesting navigators (Tab ซ้อนใน Stack)







ตัวอย่าง Nesting navigators (MyNavigator.js)

```
// MyNavigator.js
const Stack = createNativeStackNavigator();
const Tab = createBottomTabNavigator();

function MyStack() {
  return (
 <Stack.Navigator
      screenOptions={{ headerStyle: { backgroundColor: "lightblue" } }} >
      <Stack.Screen name="S1“ component={Screen1} />
      <Stack.Screen
          name="S2“ component={Screen2}
          options={({ route }) => ({
            title: route.params.prev + " ID-" + route.params.id.toString(),})}/>
       <Stack.Screen name="S3“ component={MyTab} />
  </Stack.Navigator>  );
}
```

```
// MyNavigator.js (ต่อ)
export default function MyNavigator() {
  return (
    <NavigationContainer>
      <Tab.Navigator>
        <Tab.Screen name="Tab_1" component={Tab1} />
        <Tab.Screen name="Tab_2" component={MyStack} />
        <Tab.Screen name="Tab_3" component={Tab3} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

```
//App.js
export default function App() {
  return <MyNavigator />;
}
```

ตัวอย่าง Nesting navigators (Stack ซ้อนใน Tab)