06016323 Mobile Device Programming

CHAPTER 10 : NAVIGATION (PART 2)

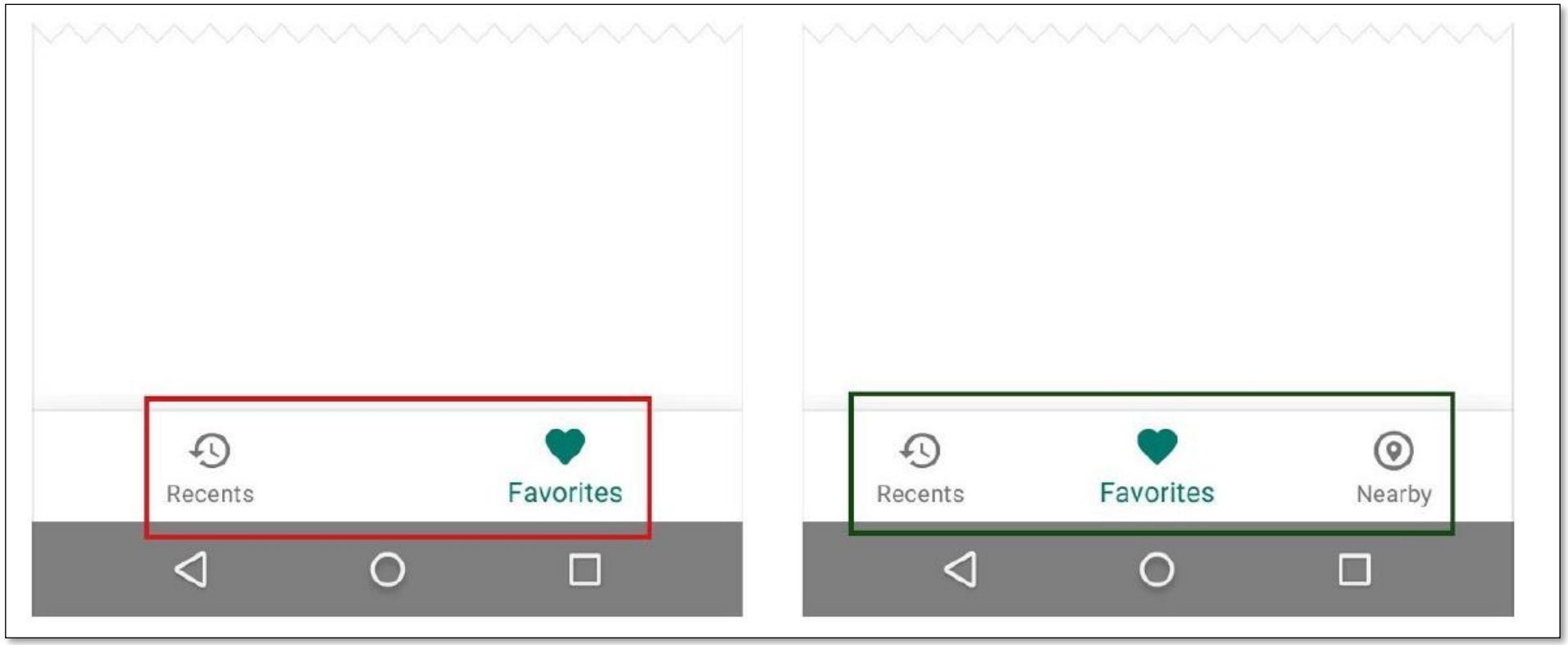
React Navigation

- Stack Navigation
- Tab Navigation
- Drawer Navigation

Tab Navigation

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

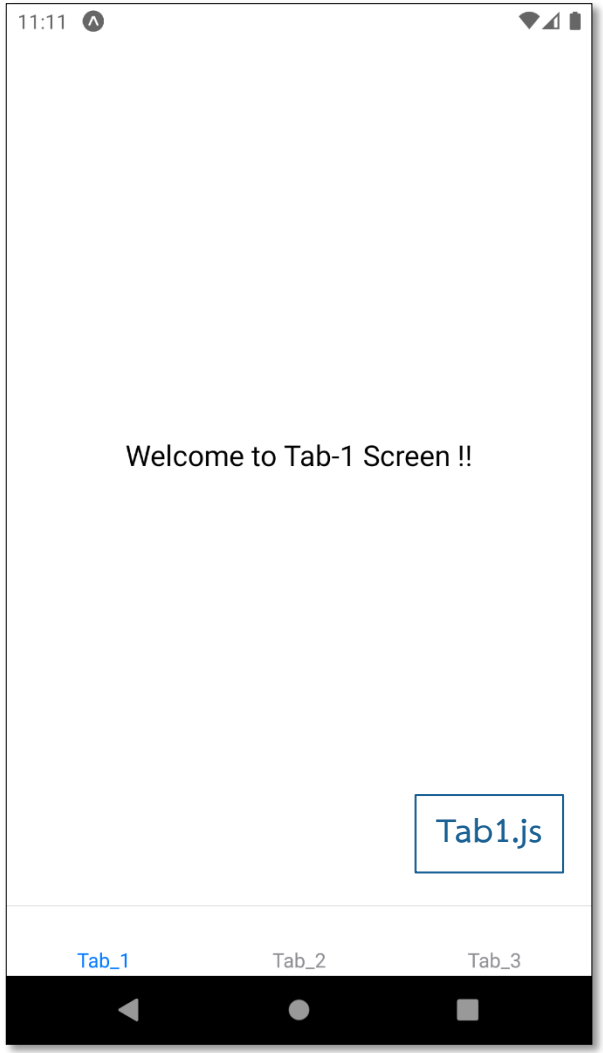
Tab Navigation

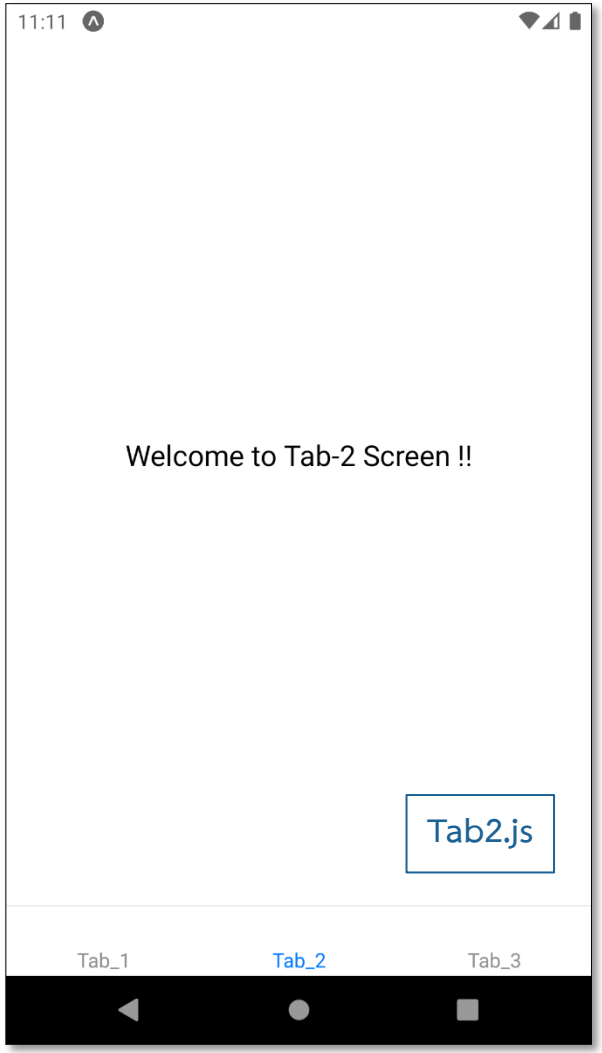


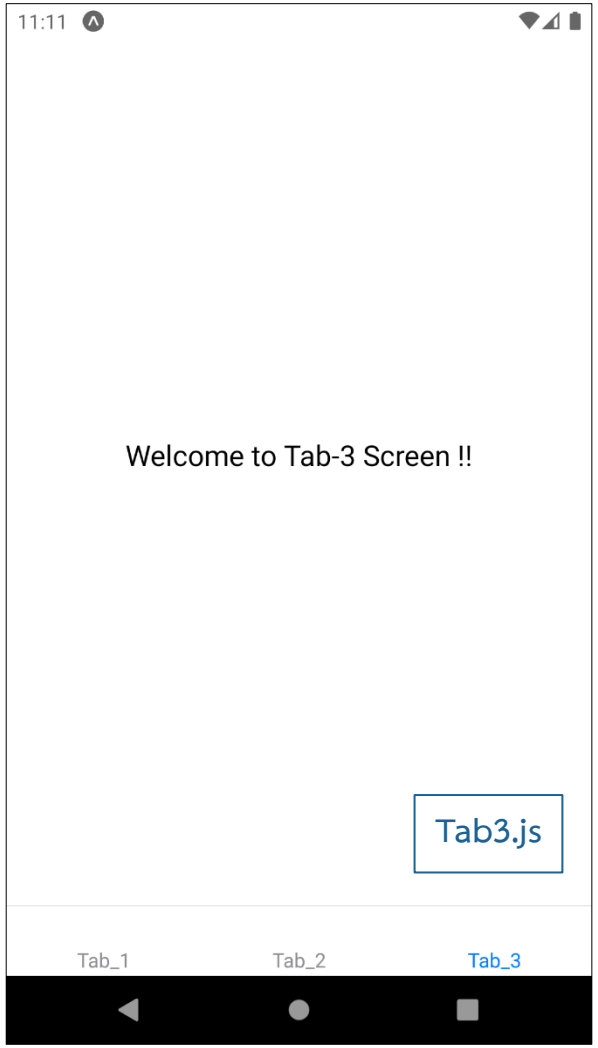
ติดตั้ง react-navigation-tabs

- สำหรับ react-navigation เวอร์ชั่น 4 ขึ้นไป ต้องมีการติดตั้งไลบรารีเพิ่มเติม เพื่อทำการสร้าง Navigator ในรูปแบบต่างๆ (ในที่นี้ จะสร้าง Tab Navigator)
  - npm install --save react-navigation-tabs หรือ
  - expo install react-navigation-tabs
- ปัจจุบัน React navigation เป็นเวอร์ชัน 5 แล้ว จะมีรูปแบบในการเขียนโปรแกรมจัดการ navigation ที่แตกต่างไป
- ศึกษาเพิ่มเติมได้ที่ : 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

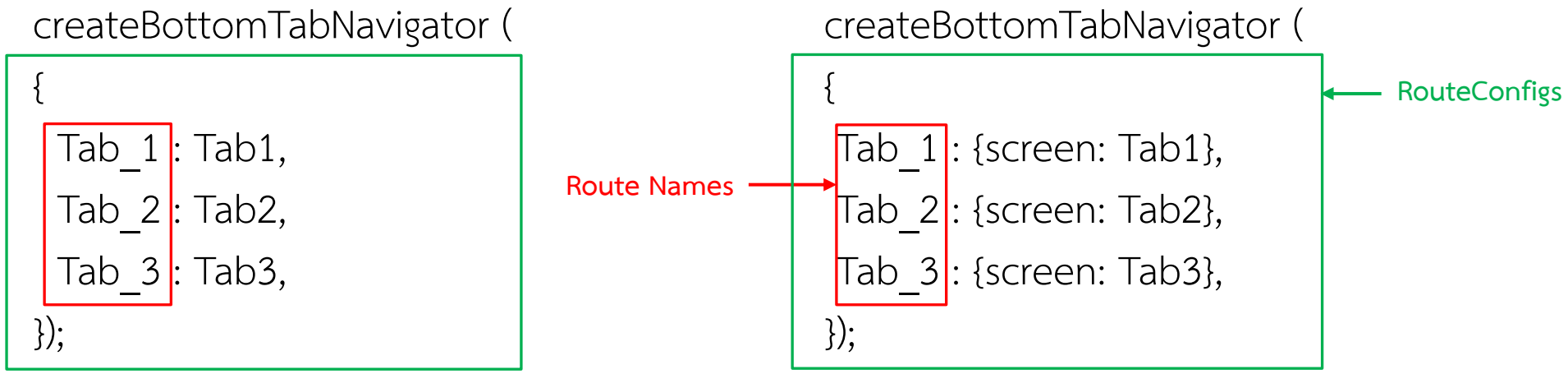
- หากต้องการสร้าง Tab navigator ทำได้โดยเรียก (คล้ายกับ Stack navigator)
  - createBottomTabNavigator
  - createMaterialBottomTabNavigator
  - createMaterialTopTabNavigator
- เริ่มต้น ต้องทำการ import สิ่งที่ใช้ในการทำ tab navigation ที่ต้องการ เช่น
  - import { createBottomTabNavigator } from "react-navigation-tabs"; // v.4
  - import { createAppContainer } from "react-navigation";
- สร้าง tab navigator ด้วย createBottomTabNavigator()
- เมื่อจะนำ Navigator ไปใช้ ต้อง export อยู่ในรูปแบบของ Navigation container ด้วย createAppContainer()

createBottomTabNavigator()

- รูปแบบ :

  createBottomTabNavigator(RouteConfigs, BottomTabNavigatorConfig);

  - คล้ายกับการเรียก createStackNavigator()



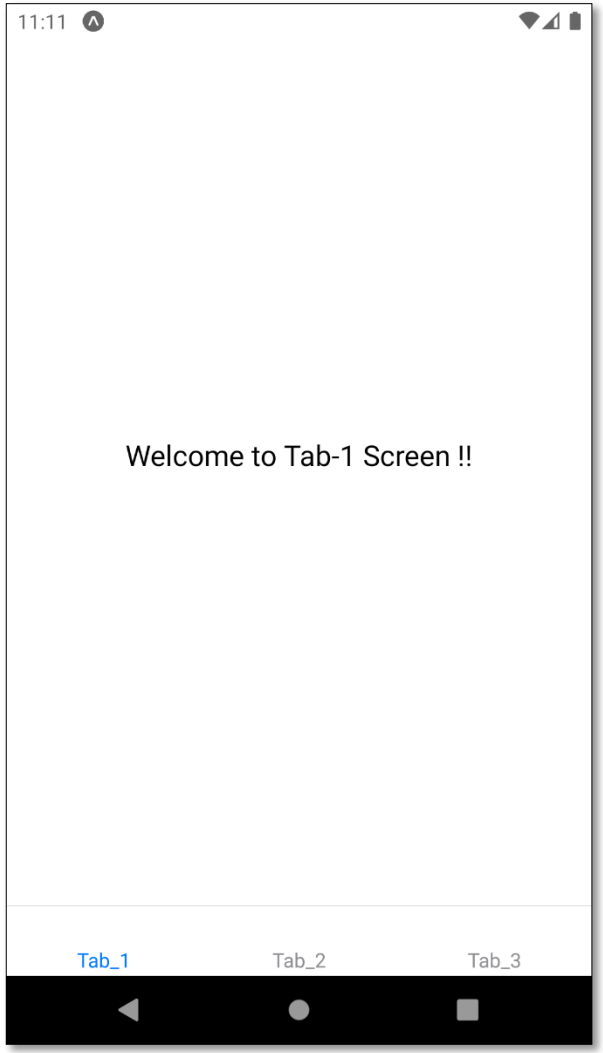
ตัวอย่างโปรแกรม MyNavigator.js

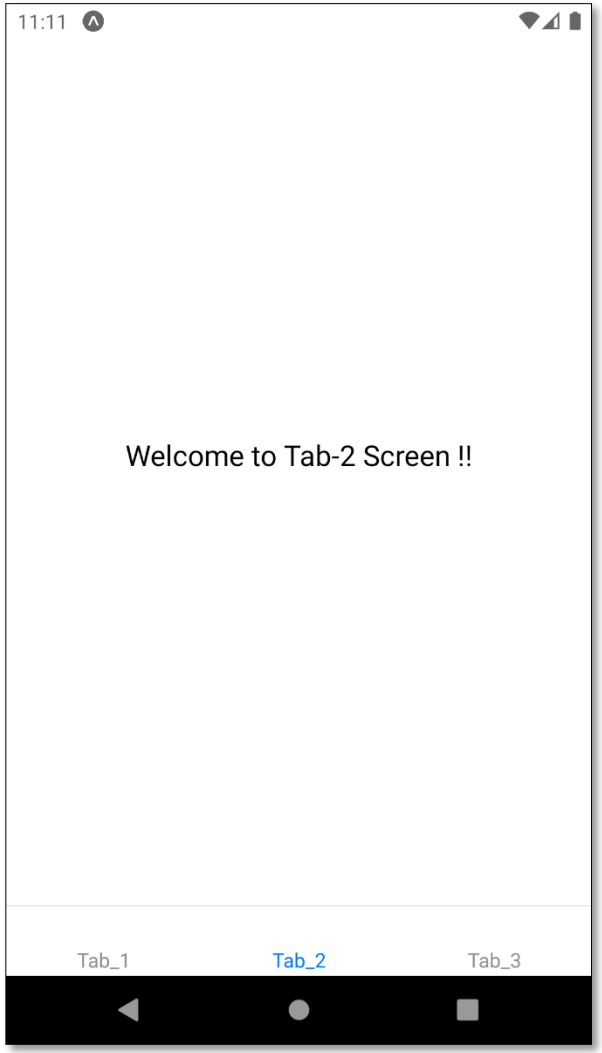
```
import { createBottomTabNavigator} from "react-navigation-tabs";
import { createAppContainer } from "react-navigation";

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

const MyTabNavigator = createBottomTabNavigator({
  Tab_1: Tab1,
  Tab_2: Tab2,
  Tab_3: Tab3,
});

export default createAppContainer(MyTabNavigator);
```

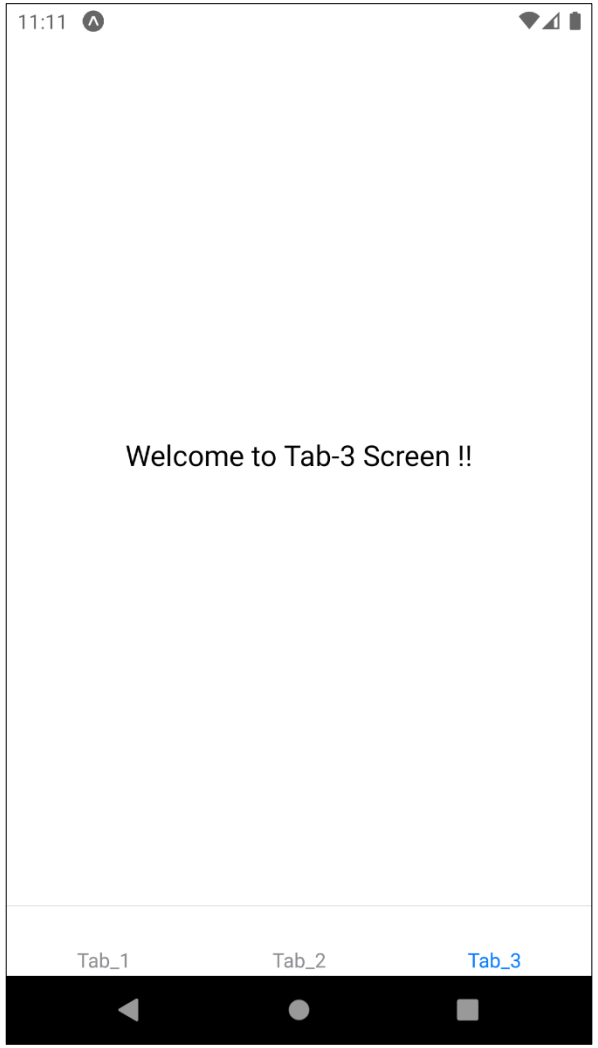
App.js

```
import MyNavigator from "./navigation/MyNavigator";

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

ตัวอย่างแอปพลิเคชัน





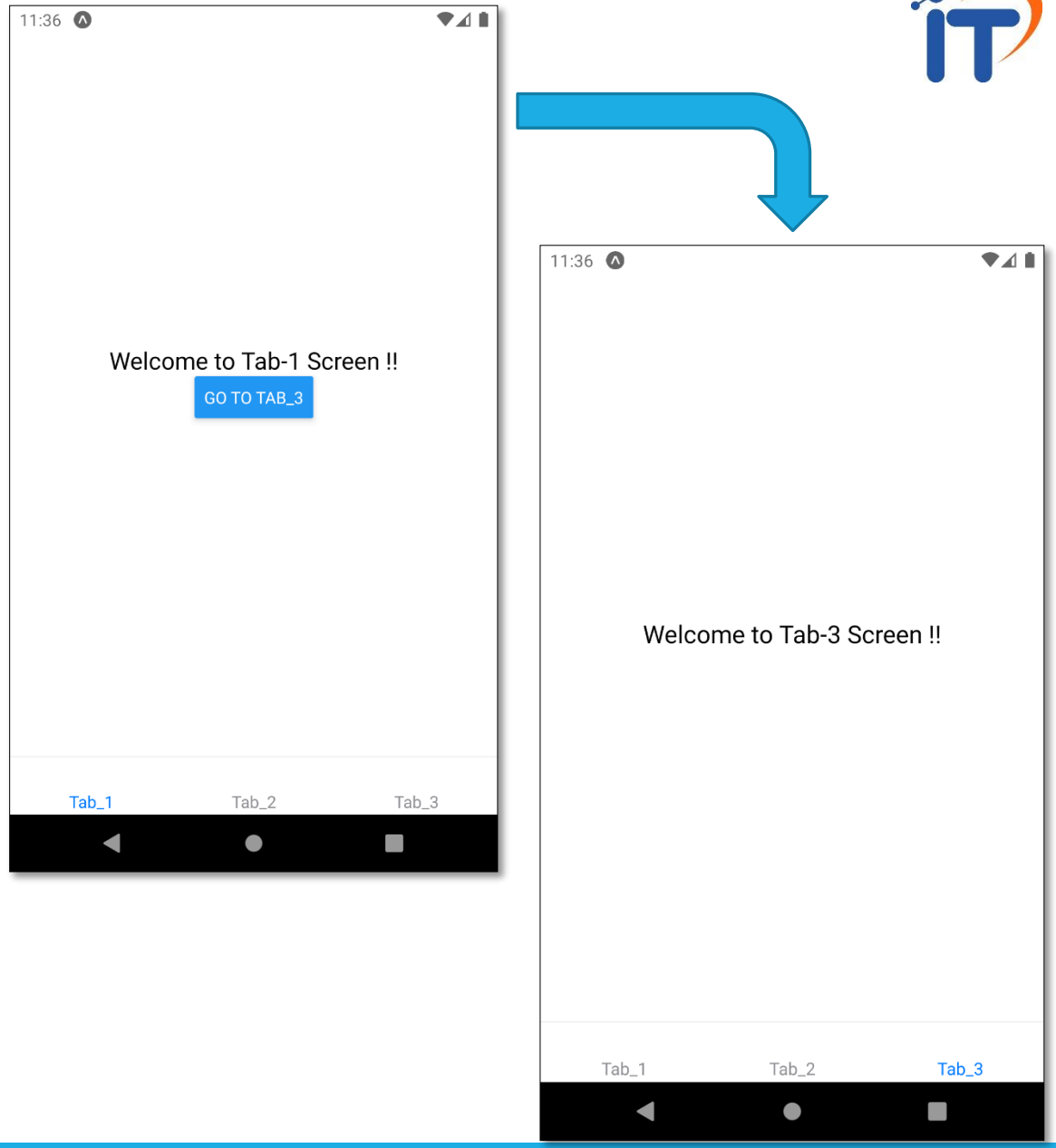


การ Navigate ระหว่าง tab

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



BottomTabNavigationConfig

- กรณีที่ต้องการปรับแต่งค่าโดยรวมของ navigation สามารถทำได้คล้ายกับ stack navigation
  - createStackNavigator(RouteConfigs, BottomTabNavigatorConfig);
  - กำหนดได้ที่อาร์กิวเมนต์ที่สองของ createBottomTabNavigator()
  - ตั้งค่าผ่าน property ที่หลากหลายได้ (ดูเพิ่มเติมใน React Navigation Docs)
    - tabBarOptions
    - initialRouteName
    - defaultNavigationOptions

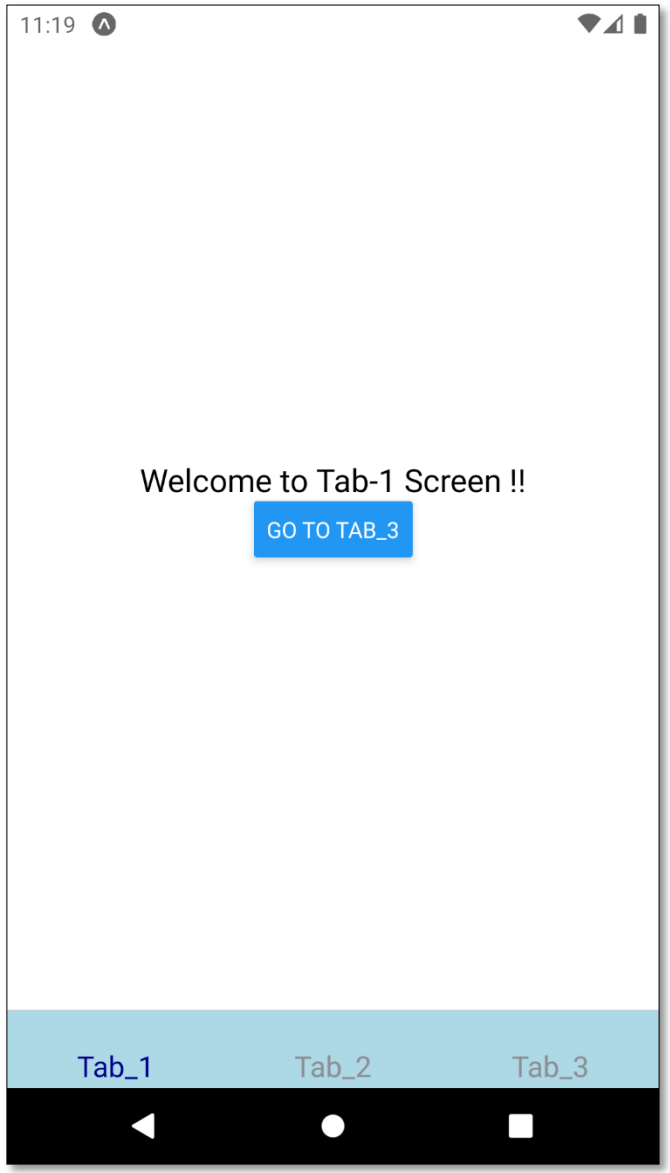
ตัวอย่างโปรแกรม MyNavigator.js

```
const MyTabNavigator = createBottomTabNavigator(
{
  Tab_1: { screen: Tab1 },
  Tab_2: { screen: Tab2 },
  Tab_3: { screen: Tab3 },
},
```

RouteConfig

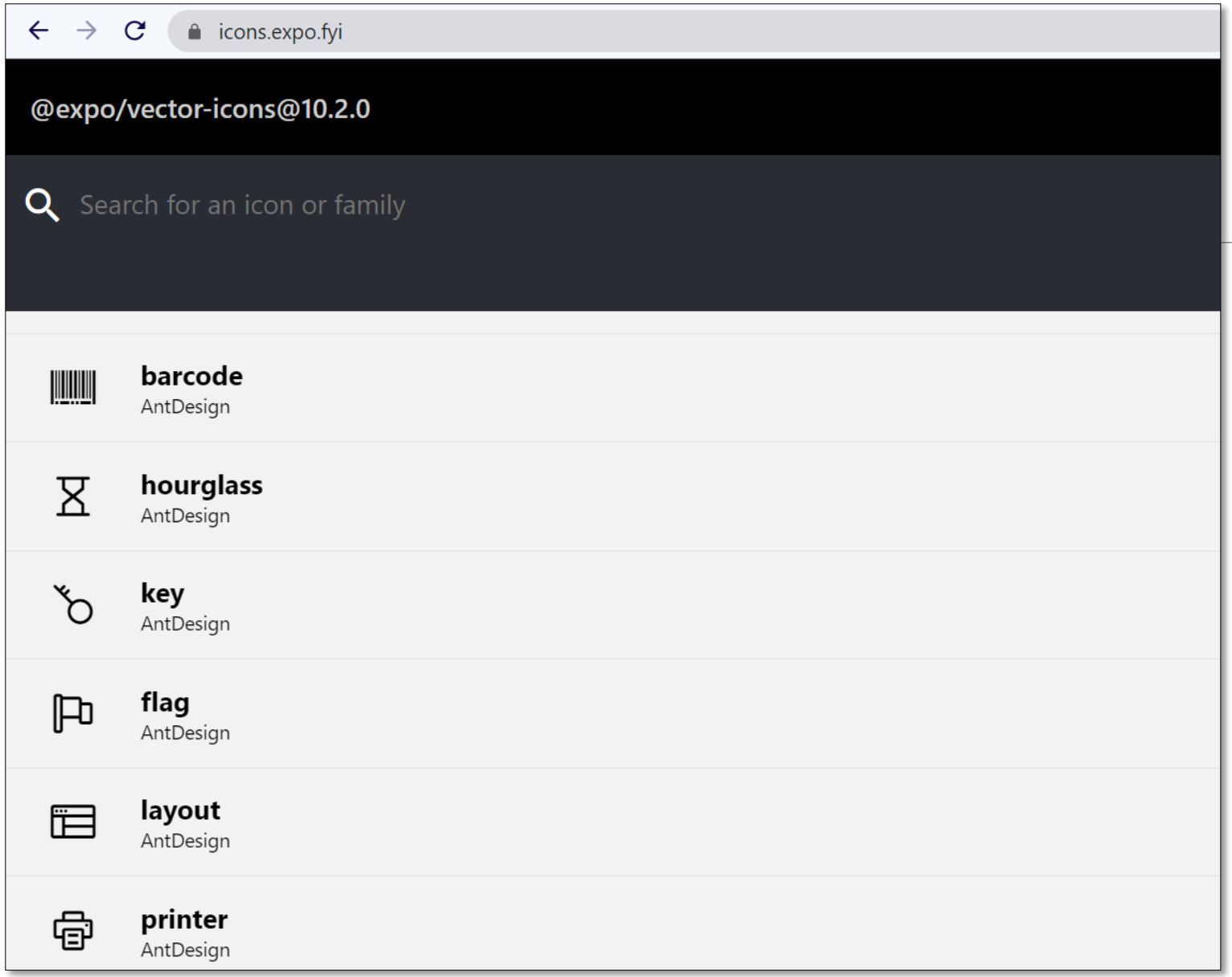
BottomTabNavigatorConfig

```
{
  tabBarOptions: {
    activeTintColor: "darkblue",
    labelStyle: { fontSize: 18, },
    style: { backgroundColor: "lightblue", },
  },
} );
```



navigationOptions

- เราสามารถปรับแต่งการตั้งค่า navigation ในแต่ละ tab ผ่าน navigationOptions เช่น
  - การกำหนดข้อความที่แสดงบน tab ผ่าน tabBarLabel
  - การกำหนดไอคอนที่ tab ผ่าน tabBarIcon
    - สามารถใช้ @expo/vector-icons ได้ (ต้องติดตั้ง @expo/vector-icons)
    - import { *icon-family* } from "@expo/vector-icons";
      - ตัวอย่าง *icon-family* เช่น Ionicons, AntDesign เป็นต้น

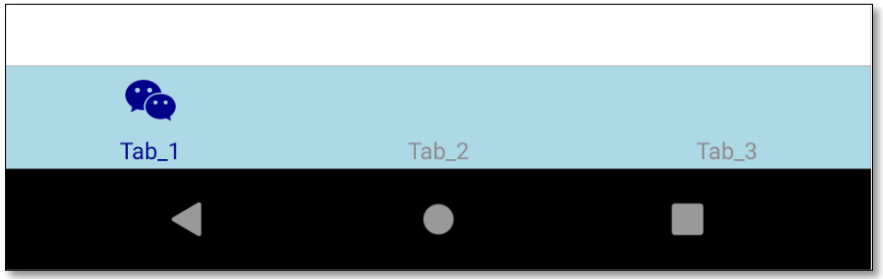


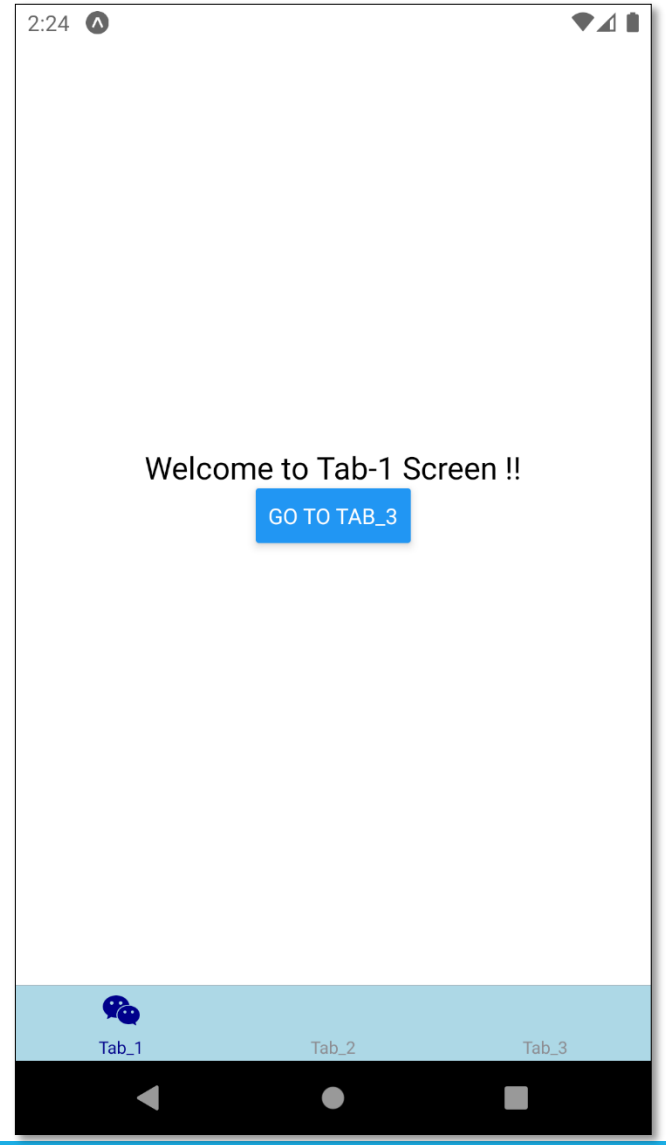
@expo/vector-icons

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

ตัวอย่างโปรแกรม MyNavigator.js

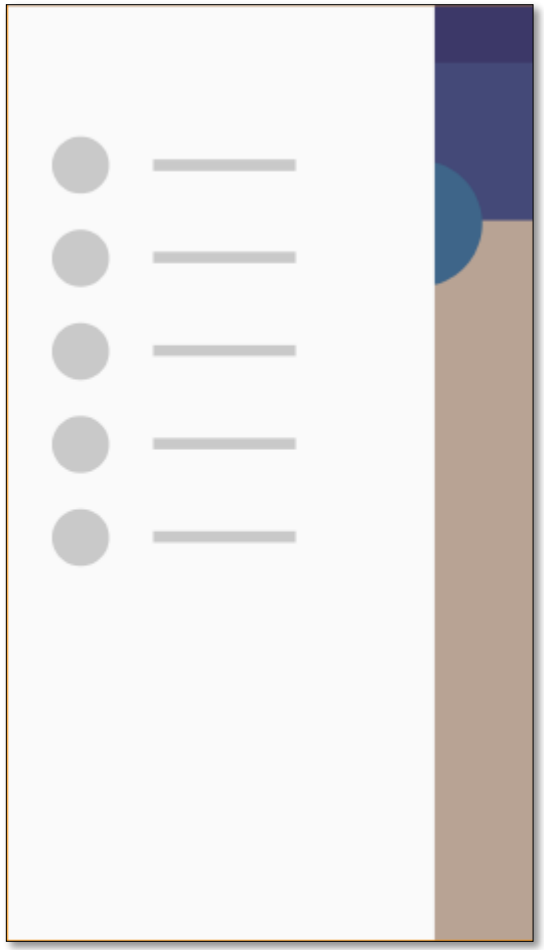
```
import React from "react";
import { AntDesign } from "@expo/vector-icons";

const MyTabNavigator = createBottomTabNavigator(
  {
    Tab_1: {
      screen: Tab1,
      navigationOptions: {
        tabBarIcon: (tabInfo) => {
          return (<AntDesign name="wechat" size={24} color={tabInfo.tintColor} />);
        }, }, },
    Tab_2: { screen: Tab2 },
    Tab_3: { screen: Tab3 },
  },
  { tabBarOptions: { ... } } );
```





Drawer Navigation

- เป็นการทำ navigation อีกรูปแบบหนึ่ง ที่มีลักษณะเป็น Slide bar จากด้านซ้ายของจอ
- install react-navigation-drawer
- import { createDrawerNavigator } from "react-navigation-drawer";
- createDrawerNavigator(RouteConfig, DrawerNavigatorConfig)
- สามารถตั้งค่า navigation ผ่าน DrawerNavigatorConfig และ navigationOptions ได้เช่นเดียวกัน (React navigation docs)



เมธอดพื้นฐานการทำ drawer navigation

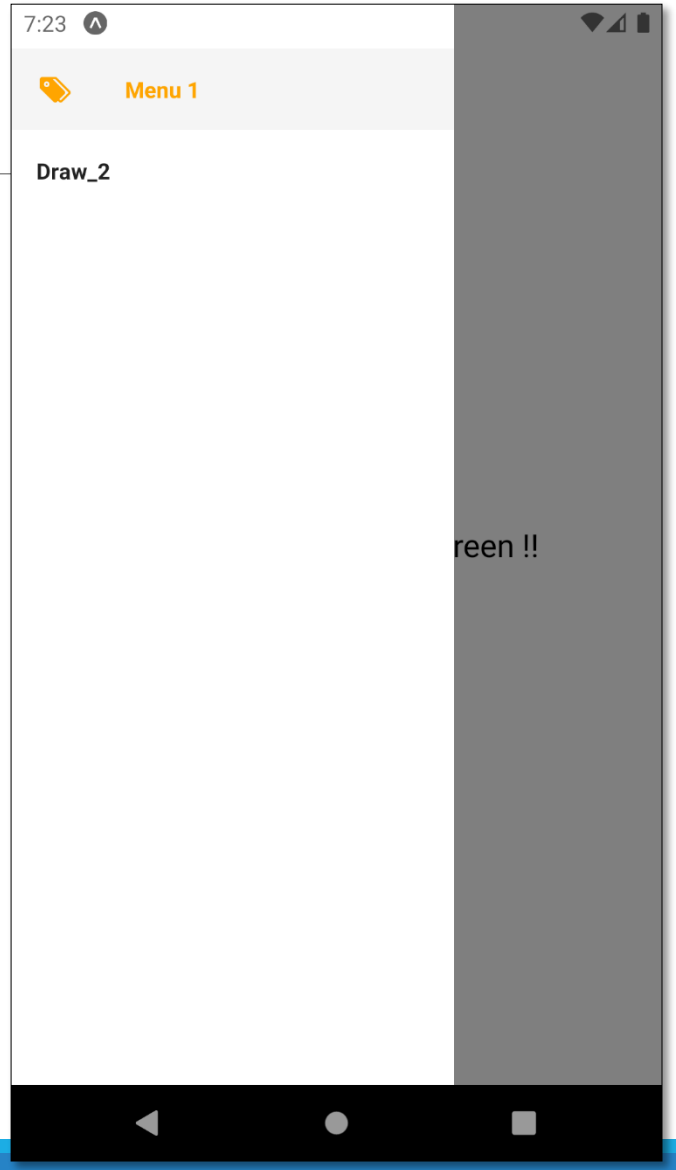
- closeDrawer
- goBack
- navigate
- openDrawer
- toggleDrawer

ตัวอย่างโปรแกรมสร้าง Drawer Navigator

```
const MyDrawerNavigator = createDrawerNavigator(
{ Draw_1: {
    screen: Draw1,
    navigationOptions: {
      drawerLabel: "Menu 1" },
      drawerIcon: () => {
        return <AntDesign name="tags" size={24} color="orange" />; }, },
  Draw_2: Draw2, },
{ contentOptions: { activeTintColor: "orange" }, }
);
```

RouteConfig

DrawerNavigatorConfig



Nesting Navigation

- เมื่อโปรแกรม/แอปพลิเคชันมีความซับซ้อน โดยมีหน้าจอหลายส่วนและจำเป็นต้องใช้รูปแบบ navigation หลายรูปแบบ
- ในกรณีนี้ เราสามารถใช้ Nesting navigation ได้ โดยทำการซ้อน navigator หนึ่งอยู่ใน navigator อื่นได้
  - เช่น การซ้อน tab navigator ใน stack navigator เป็นต้น

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

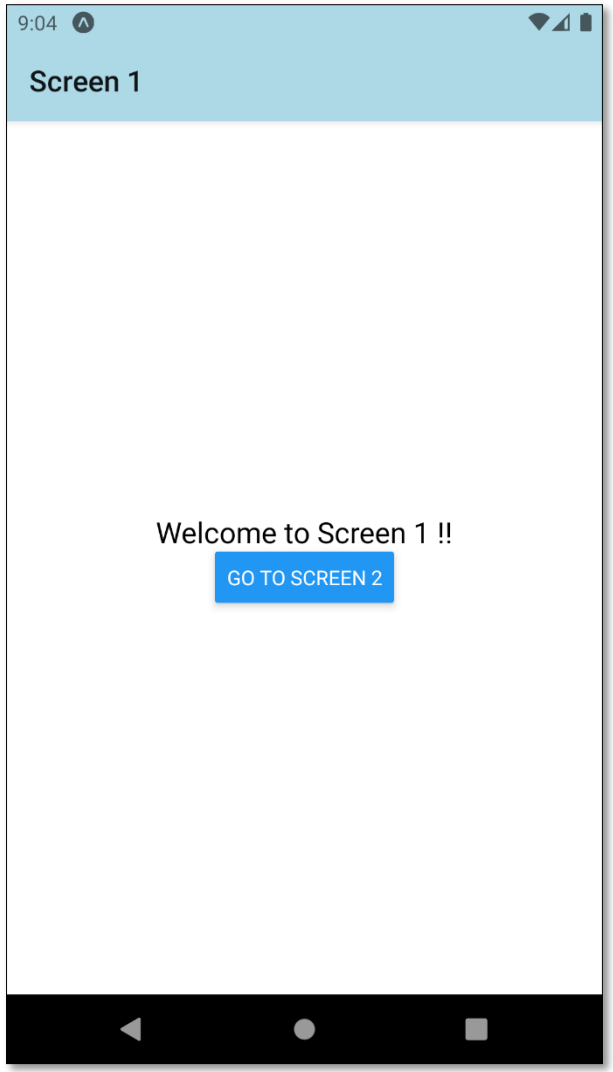
- Navigation action จะถูกจัดการด้วย navigator ล่าสุด ถ้าไม่สามารถจัดการได้ navigator ภายนอกจะพยายามจัดการเอง
- Navigator ที่อยู่ภายใน จะสามารถใช้เมธอดเฉพาะของ navigator ภายนอกได้
  - เช่น tab ซ้อนใน stack navigator แล้ว หน้าจอของ tab navigator นั้นจะมีเมธอด push และ replace ใน navigation prop ด้วย
- 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

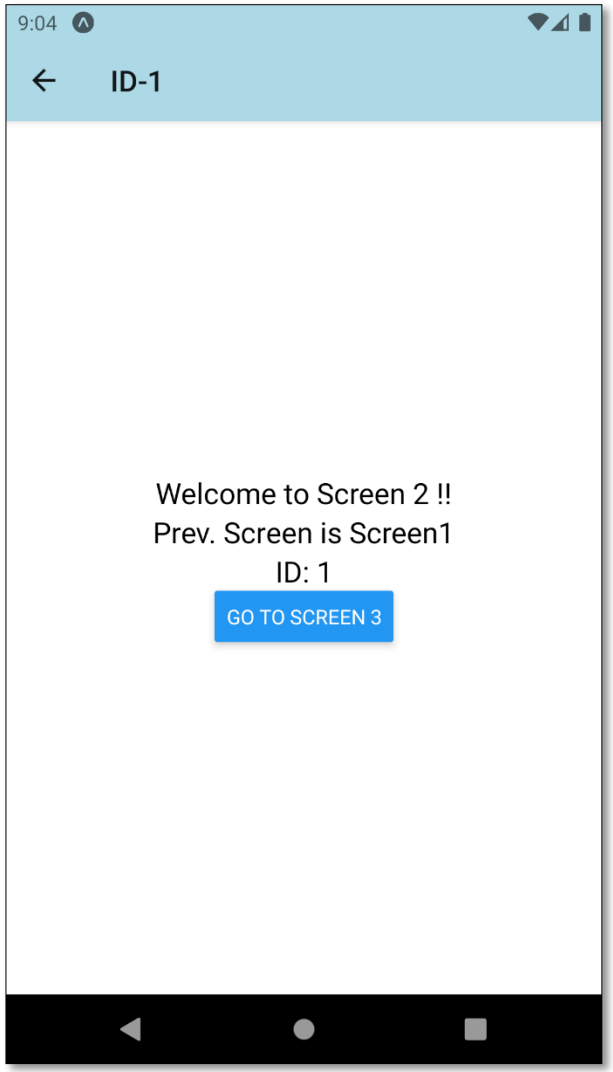
ตัวอย่าง Nesting navigators (MyNavigator.js)

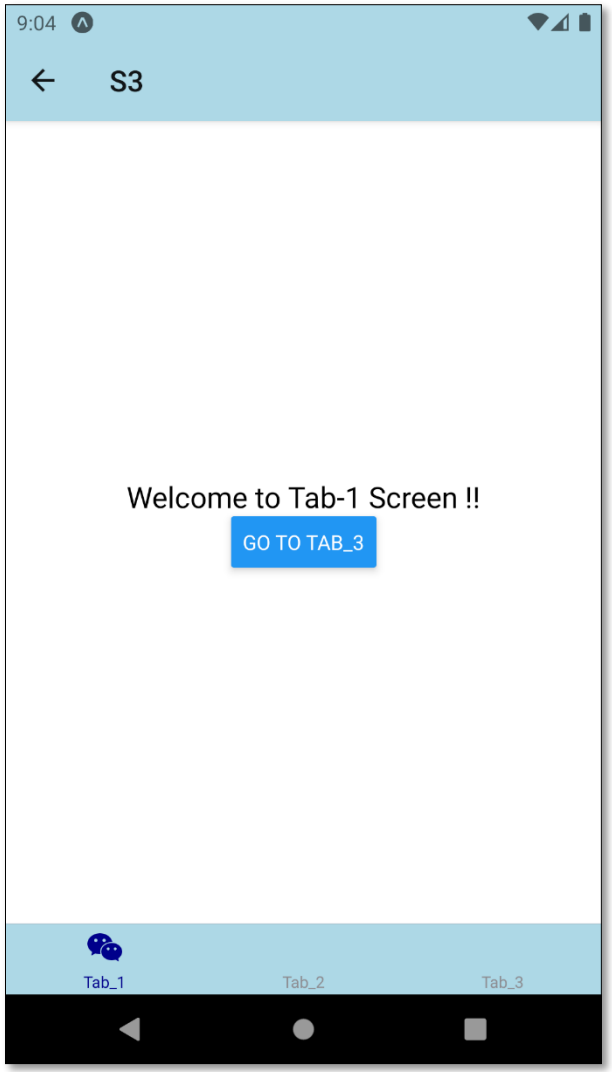
```
const MyTabNavigator = createBottomTabNavigator(
  {
    Tab_1: {
      screen: Tab1,
      navigationOptions: { ... } },
    Tab_2: { screen: Tab2 },
    Tab_3: { screen: Tab3 },
  },
  { tabBarOptions: { ... } } );
```

```
const MyStackNavigator = createStackNavigator(
  {
    S1: {
      screen: Screen1,
      navigationOptions: { ... } },
    S2: { screen: Screen2 },
    S3: MyTabNavigator,
  },
  { defaultNavigationOptions: { ... } } );

export default
createAppContainer(MyStackNavigator);
```

ตัวอย่าง Nesting navigators (Tab ซ้อนใน Stack)





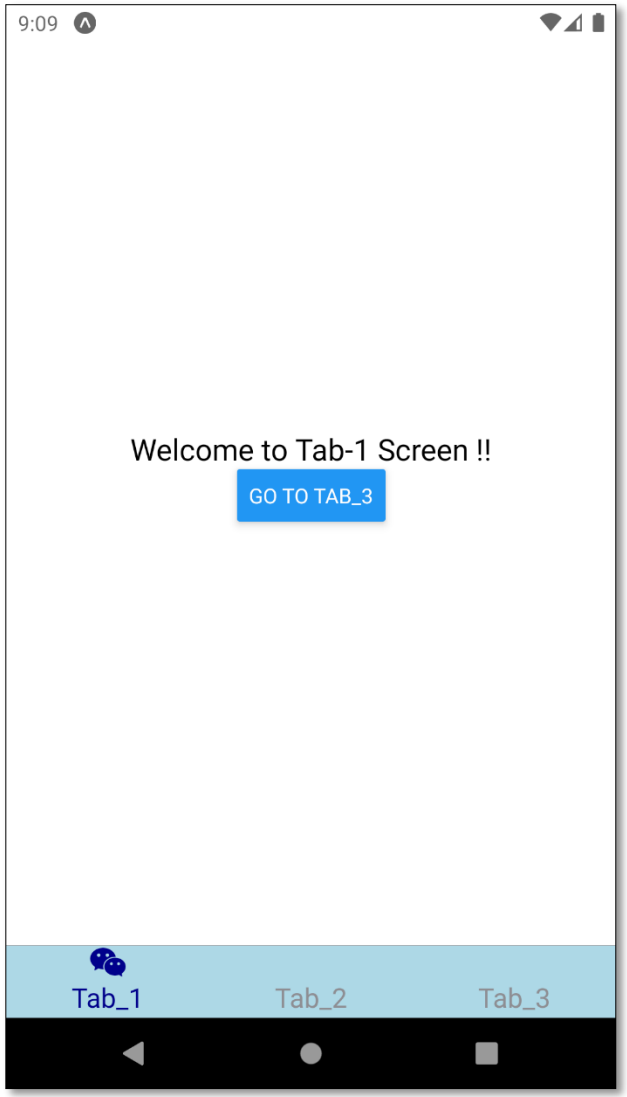


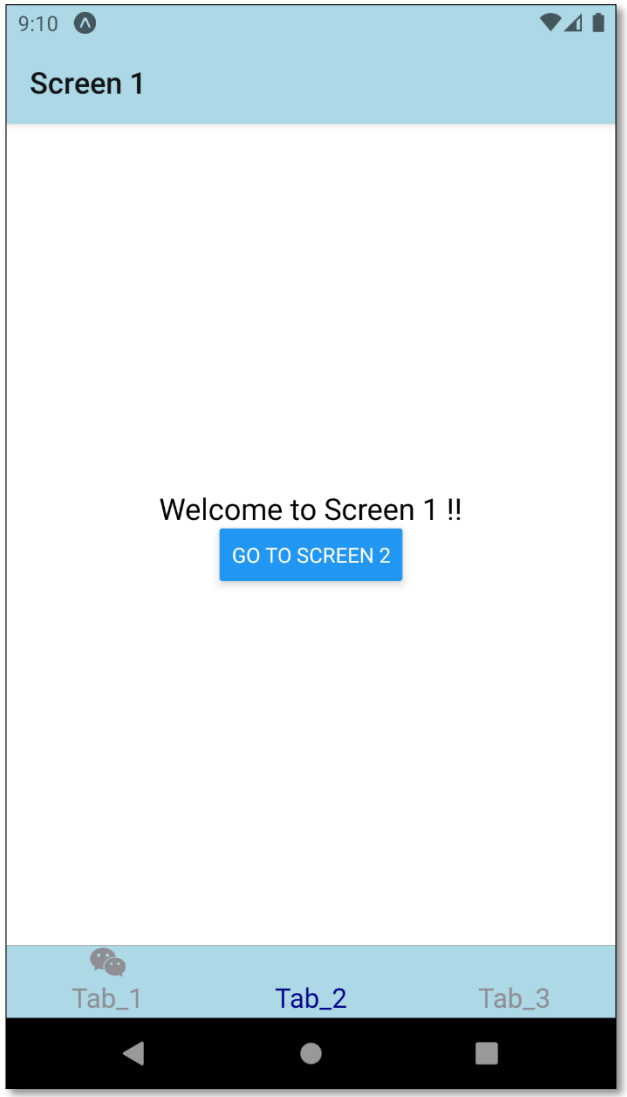
ตัวอย่าง Nesting navigators (MyNavigator.js)

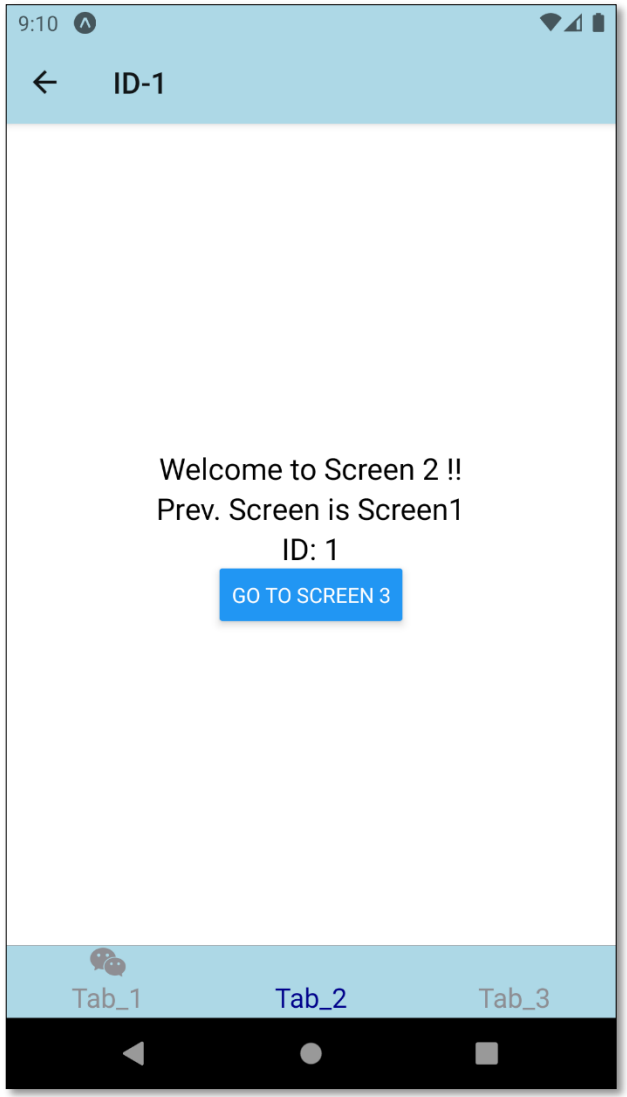
```
const MyStackNavigator = createStackNavigator(
 {
   S1: {
     screen: Screen1,
     navigationOptions: { ... } },
   S2: { screen: Screen2 },
   S3: { screen: Screen3 },
 },
 { defaultNavigationOptions: { ... } } );
```

```
const MyTabNavigator = createBottomTabNavigator(
 {
   Tab_1: {
     screen: Tab1,
     navigationOptions: { ... } },
   Tab_2: MyStackNavigator,
   Tab_3: { screen: Tab2 },
 },
 { tabBarOptions: { ... } } );

export default createAppContainer(MyTabNavigator);
```

ตัวอย่าง Nesting navigators (Stack ซ้อนใน Tab)





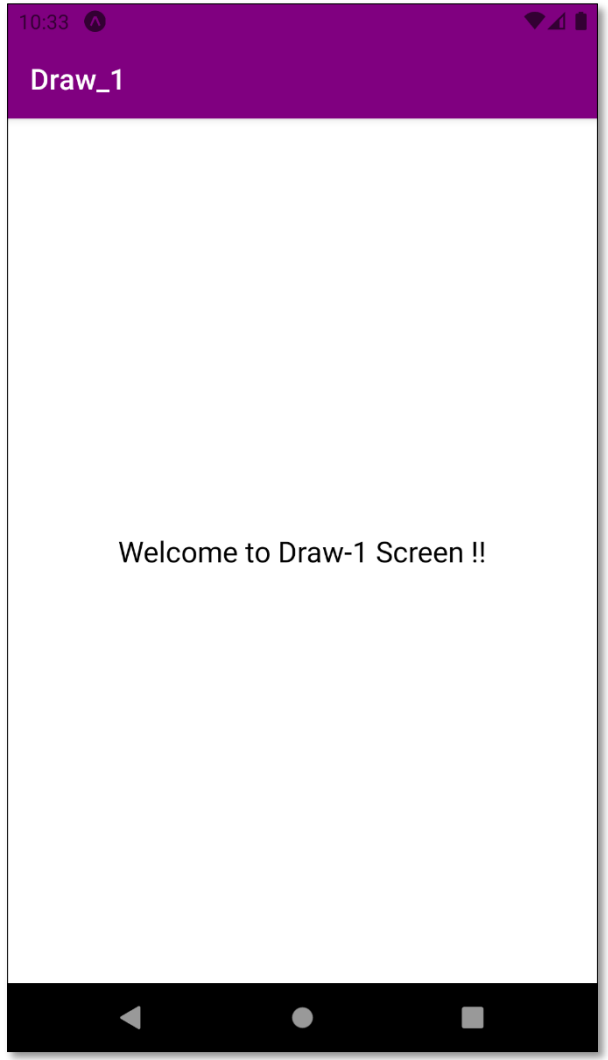


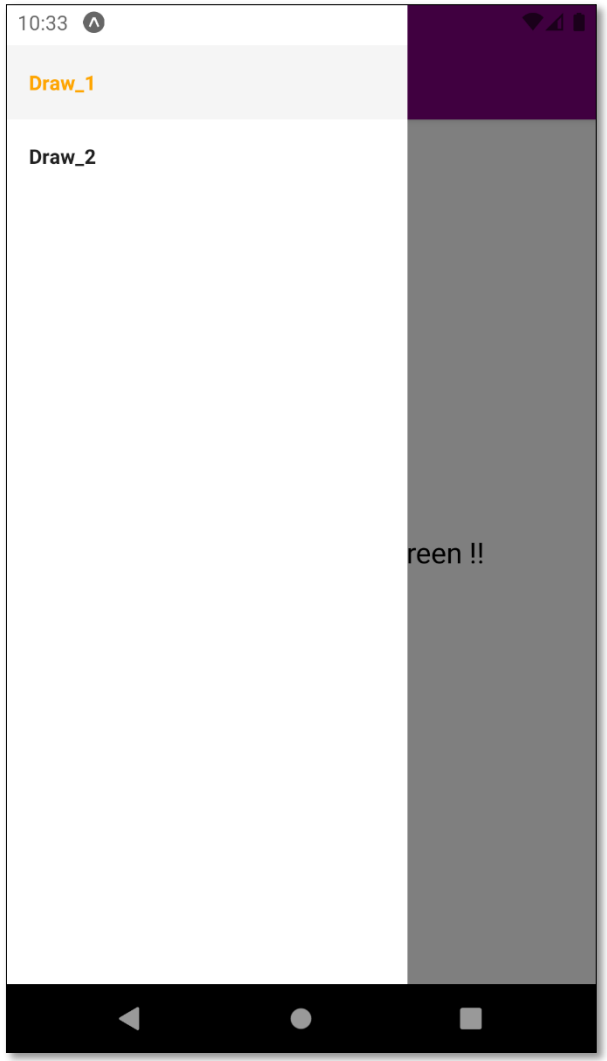
ตัวอย่าง Nesting navigation (ซ้อน stack เข้าใน drawer)

```
const MyStackDraw1Nav = createStackNavigator(
  {
    Draw_1: { screen: Draw1 },
    ...
  },
  { defaultNavigationOptions: { ... } } );
```

```
const MyDrawerNavigator = createDrawerNavigator(
  {
    Draw_1 : MyStackDraw1Nav,
    ...
  },
  { contentOptions: { ... } } );

export default
createAppContainer(MyDrawerNavigator);
```

ตัวอย่าง Nesting navigators (Stack ซ้อนใน Drawer)





- สังเกตว่า การเปิด Drawer จะต้องลากจากด้านข้างจอ
- ในกรณีที่ต้องการเปิด Drawer จากปุ่มที่แถบเฮดเดอร์ สามารถทำได้โดยใช้ Header Button

การสร้างปุ่มบริเวณเฮดเดอร์ของหน้าจอ

- ติดตั้ง react-navigation-header-buttons
- สร้างคอมโพเนนต์เพื่อใช้ในการสร้างปุ่ม (HeaderButton) เช่น MyHeaderButton.js
- ปรับปรุงโปรแกรมในส่วนของหน้าจอที่ต้องการแสดงปุ่ม เช่น Draw1.js
  - ทำการกำหนด navigationOptions ของหน้าจอนั้น
  - กำหนด property headerLeft
    - รีเทิร์น <HeaderButtons> โดยกำหนดลักษณะของ HeaderButton ที่จะแสดง

MyHeaderButton.js

```
import React from "react";
import { Platform } from "react-native";
import { HeaderButton } from "react-navigation-header-buttons";
import { Ionicons } from "@expo/vector-icons";

const MyHeaderButton = (props) => {
  return (
    <HeaderButton
        {...props}
        IconComponent={Ionicons}
        iconSize={23}
        color={Platform.OS === "android" ? "orange" : "purple"}
    />
  ); };

export default MyHeaderButton;
```

Draw1.js

```
import { HeaderButtons, Item } from
      "react-navigation-header-buttons";
import MyHeaderButton from
      "../components/MyHeaderButton";

const Draw1 = (props) => {
  return (
    <View>
      <Text>Welcome to Draw-1 Screen !!</Text>
    </View>
  );
};
```

... ต่อด้านขวา ...

```
Draw1.navigationOptions = (navigationData) => {
  return {
    headerTitle: "Draw1-Screen",
    headerLeft: () => {
      return (
        <HeaderButtons HeaderButtonComponent={MyHeaderButton}>
          <Item
            title="Menu"
            iconName="ios-list"
            onPress={() => {
              navigationData.navigation.toggleDrawer();
            }}
          />
        </HeaderButtons>
      ); }, }; };
```

ตัวอย่างการสร้าง header button