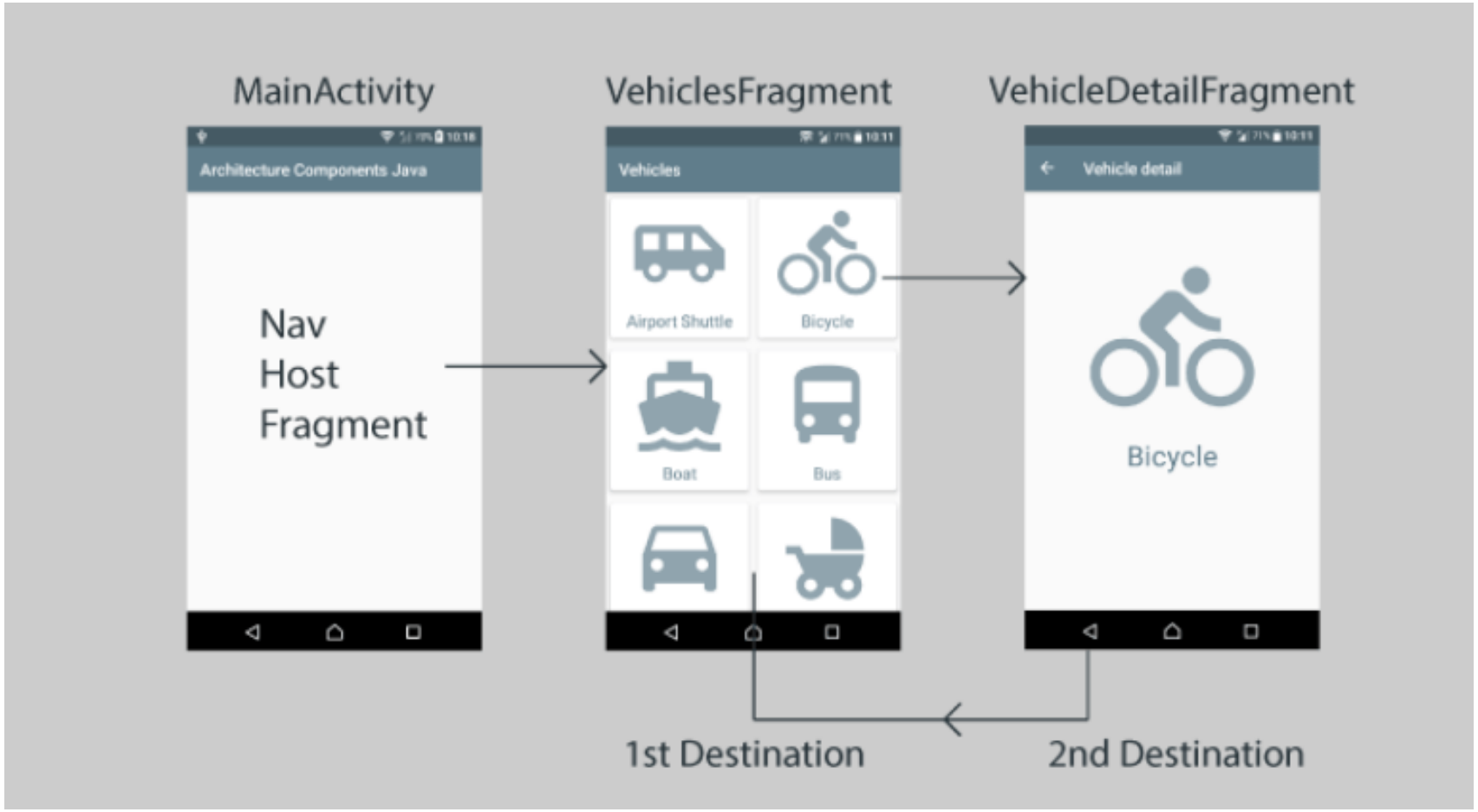
06016323 Mobile Device Programming

CHAPTER 5 : NAVIGATION (PART 1)

Navigation

- แอปพลิเคชันประกอบไปด้วยส่วนประกอบต่างๆ ซึ่งสามารถแสดงในรูปของหน้าจอ (Screen) ที่แตกต่างกัน
- Navigation เป็นส่วนสำคัญที่ช่วยจัดการการเปลี่ยนหน้าจอไปมา รวมถึงการจดจำลำดับการเปิดหน้าจอที่ผ่านมาอีกด้วย

Navigation



ที่มา : https://developersbreach.com/navigation-with-architecture-components-android/

Native Navigation

- รูปแบบการจัดการ Navigation ในแอปพลิเคชัน ของแต่ละแพลตฟอร์ม จะถูกจัดการด้วย Native component ที่ต่างกัน
  - iOS : UINavigationController
  - Android : Activities
- React Native : จะทำงานบน JavaScript thread และจะมี Main thread ที่ทำหน้าที่ render หน้าจอที่เป็น Native views ของทั้ง iOS และ Android

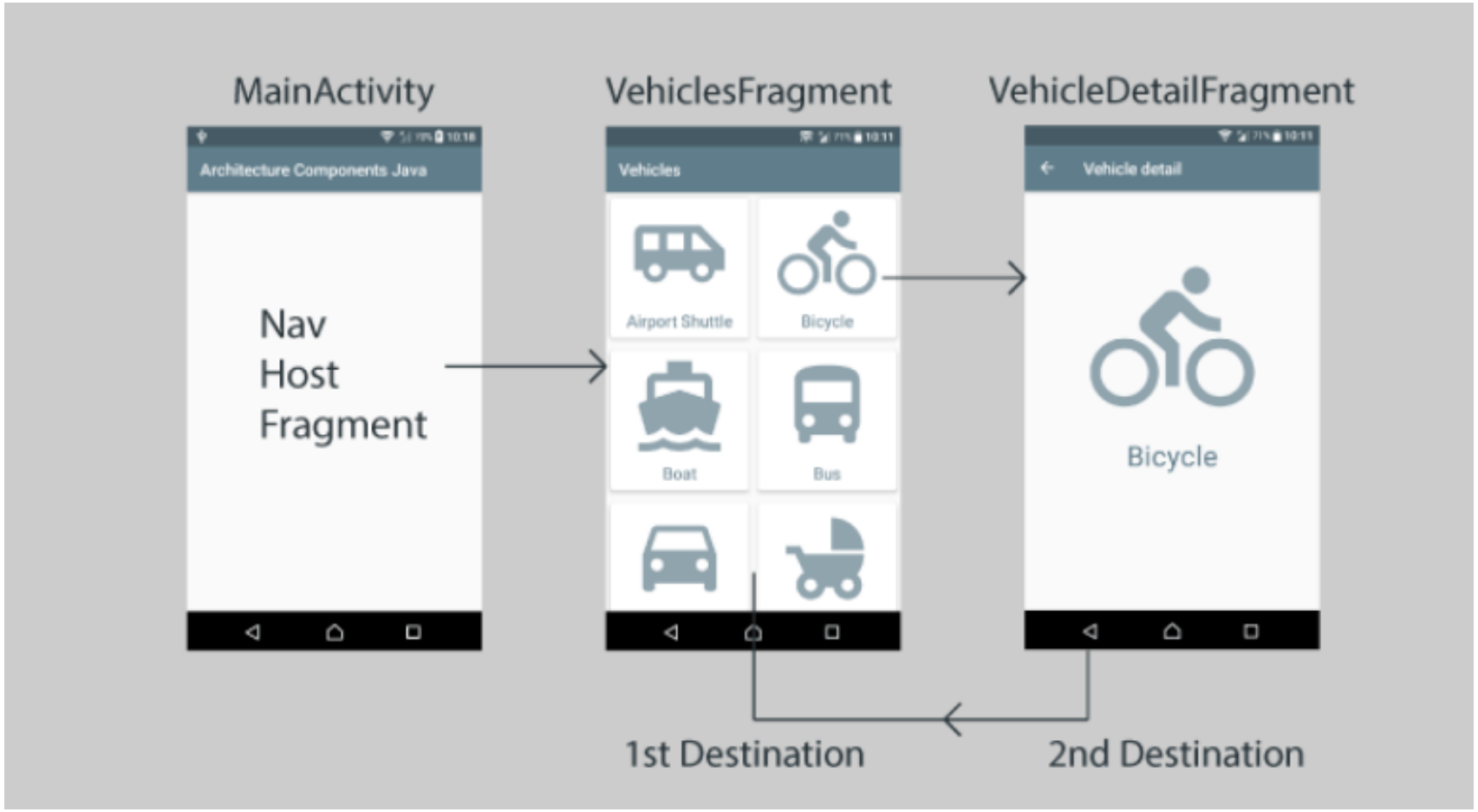
React Navigation

- Stack Navigation
- Tab Navigation
- Drawer Navigation

Stack Navigation

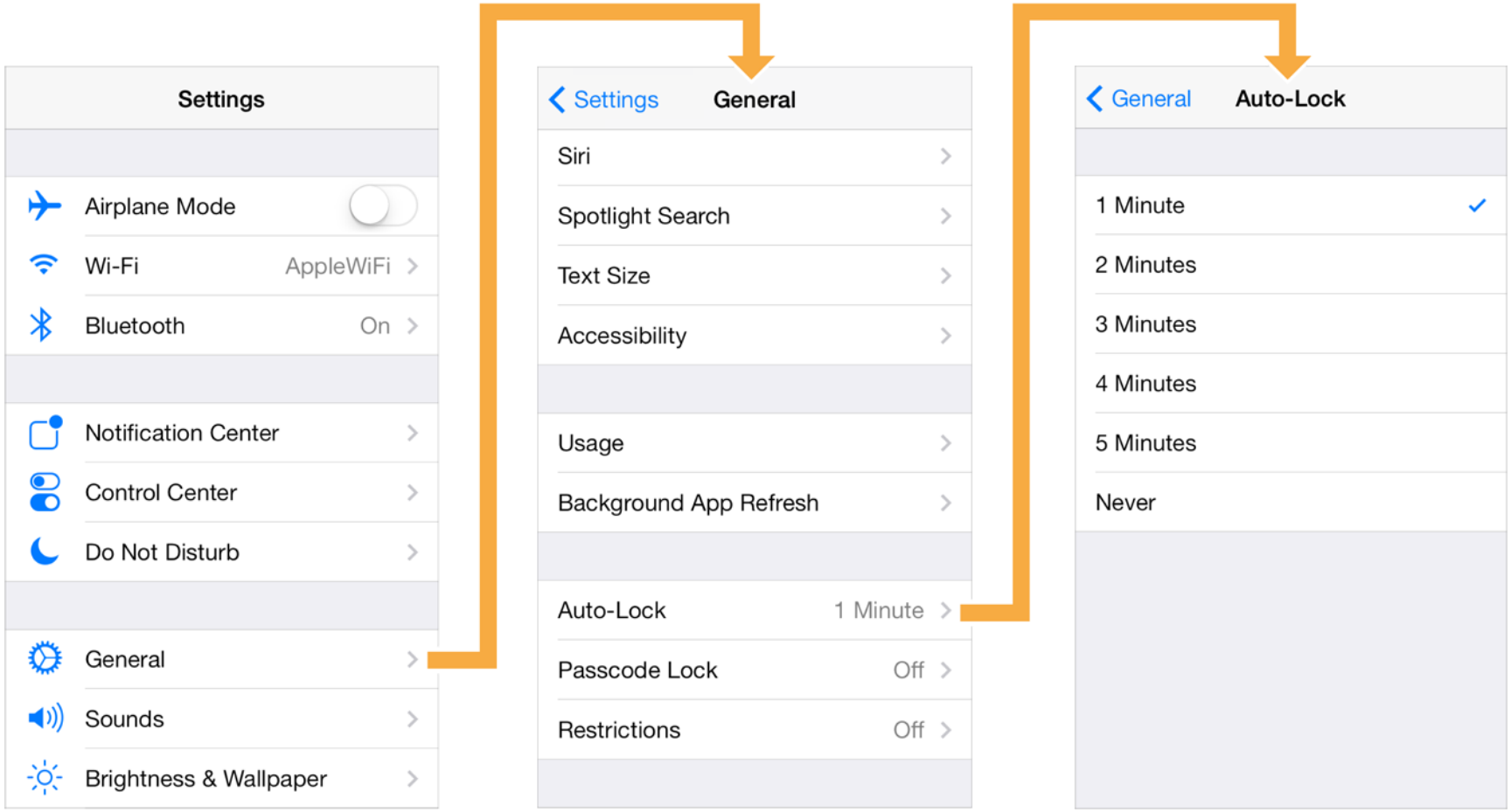
- การเปลี่ยนไปยังหน้าจออื่น Stack navigator จะทำการ push หน้าจอใหม่เข้าไปใน Navigation stack (อยู่ที่ส่วน top ของ stack)
- เมื่อมีการย้อนกลับไปหน้าจอก่อนหน้า Stack navigator จะทำการ pop หน้าจอล่าสุดออก
- หน้าจอที่ตำแหน่ง top ของ stack จะถูกแสดงให้ผู้ใช้เห็น ณ ขณะนั้น

Stack Navigation (Android)



ที่มา : https://developersbreach.com/navigation-with-architecture-components-android/

Stack Navigation (iOS)



ที่มา : https://developer.apple.com/documentation/uikit/uinavigationcontroller

การเขียนโปรแกรมเพื่อจัดการ Navigation ใน React Native

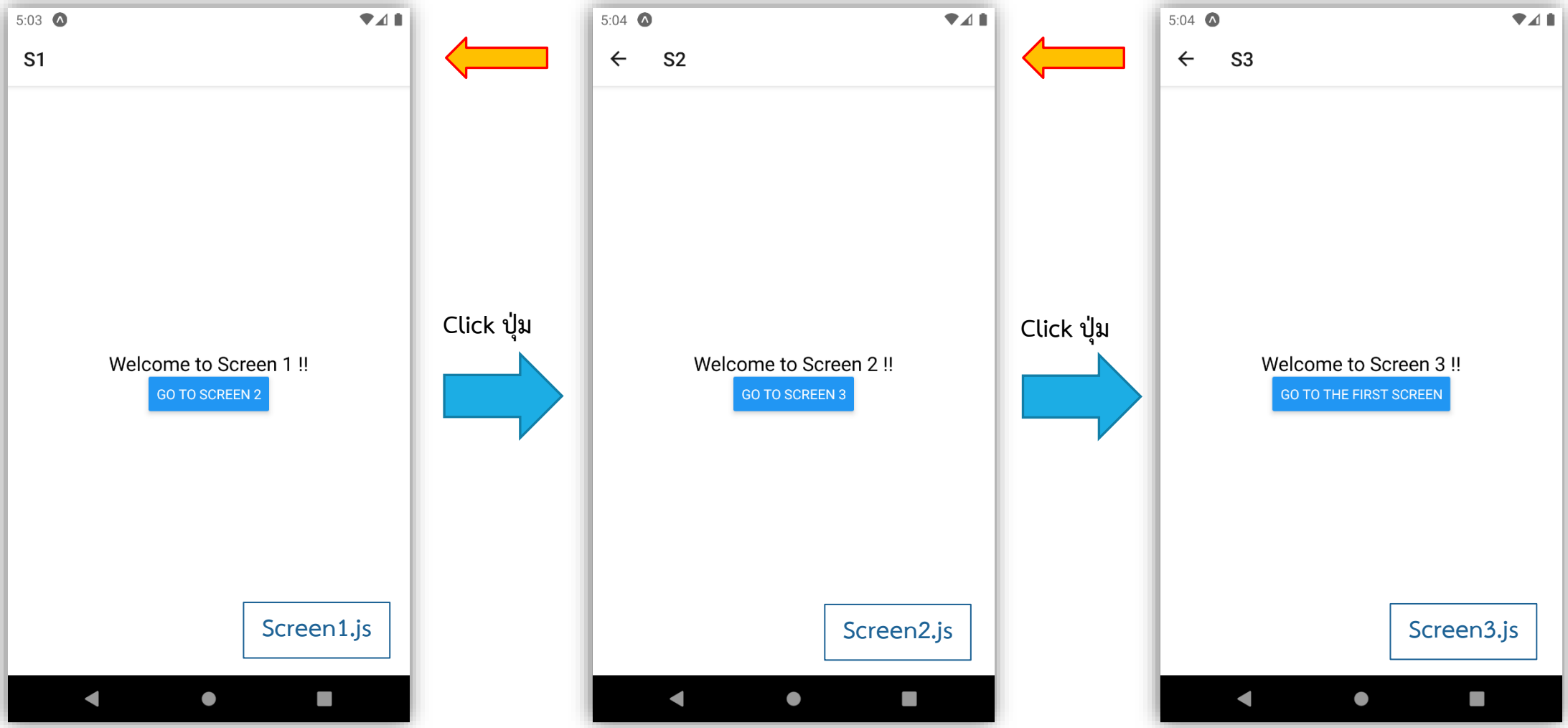
- ติดตั้งไลบรารี React Navigation เพื่อใช้จัดการการทำ Navigation
  - npm install @react-navigation/native หรือ
  - expo install @react-navigation/native
- ติดตั้ง dependencies ที่ React navigation ต้องการเพิ่มเติม
  - expo install react-native-screens react-native-safe-area-context

** React Navigation ในเอกสารนี้ เป็น React Navigation version 6.x

ติดตั้ง native stack navigation

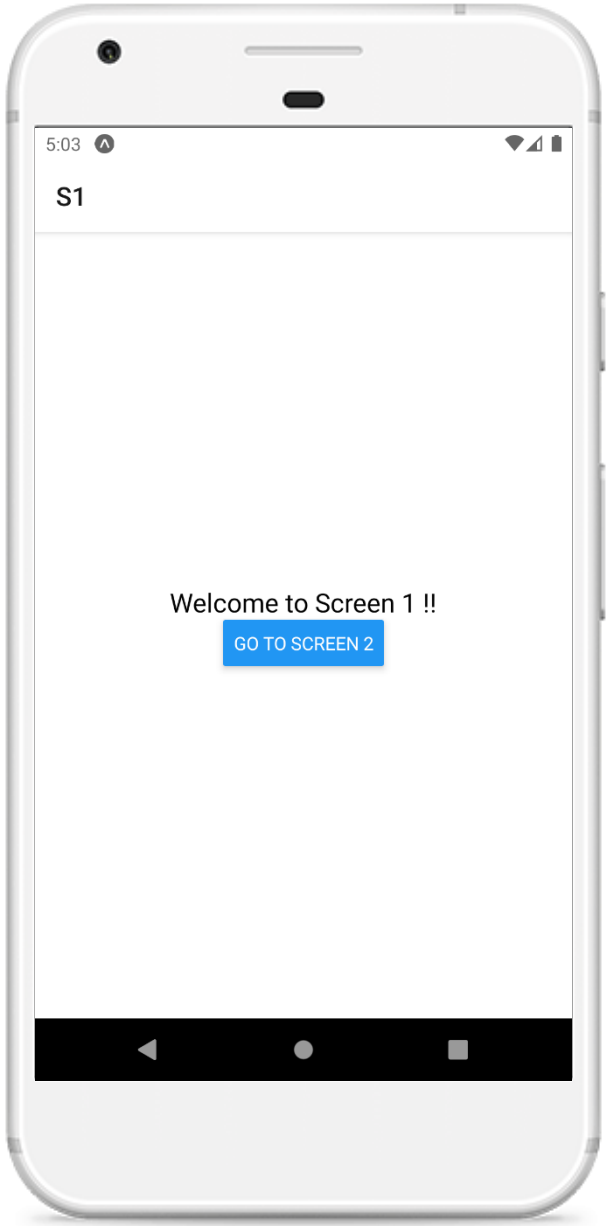
- สำหรับ react-navigation เวอร์ชั่น 6 ขึ้นไป ต้องมีการติดตั้งไลบรารีเพิ่มเติม เพื่อทำการสร้าง Navigator ในรูปแบบต่างๆ (ในที่นี้ จะสร้าง Stack Navigator)
  - npm install @react-navigation/native-stack หรือ
  - expo install @react-navigation/native-stack

ตัวอย่างแอปพลิเคชัน



ตัวอย่างโปรแกรม Screen1.js

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

const Screen1 = () => {
  return (
    <View>
      <Text>Welcome to Screen 1 !!</Text>
      <Button
        title="Go to Screen 2"
        onPress={ () => { } }
      />
    </View>
  );
};
export default Screen1;
```



การสร้าง Native Stack Navigator

- หากต้องการทำ Stack Navigation ในแอปพลิเคชัน ต้องทำการสร้าง Navigator ที่ทำหน้าที่กำหนด navigation logic ขึ้นมาก่อน
  - import { NavigationContainer } from '@react-navigation/native'; // v.6.x
  - import { createNativeStackNavigator } from '@react-navigation/native-stack';
  - สร้าง Stack navigator ด้วยเมธอด createNativeStackNavigator()
  - ทำการกำหนด Navigation logic ผ่าน Stack navigator ที่สร้างขึ้นข้างต้น

createNativeStackNavigator

- เป็นฟังก์ชันที่ใช้สร้าง Stack navigator ซึ่งจะคืนค่าอ็อบเจคต์ที่มี property 2 อย่าง
  - 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

NavigationContainer

- เป็นคอมโพเนนต์ที่ใช้จัดการลำดับการทำ navigation
- ปกติแล้ว เราจะเรนเดอร์คอมโพเนนต์นี้ที่ส่วน root ของแอปพลิเคชัน (App.js)

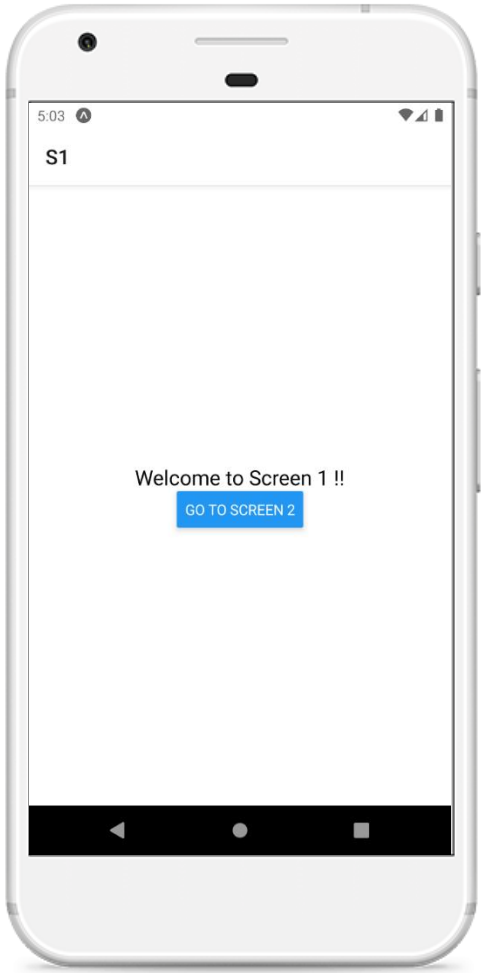
การกำหนด Stack navigation ในโปรแกรม

- สร้าง Native stack navigator (createNativeStackNavigator)
- คืนค่าคอมโพเนนต์ NavigationContainer (ในส่วนของ function App())
- ภายใน NavigationContainer ให้เรียกคอมโพเนนต์ Navigator ของ stack ที่สร้างขึ้น
- ภายในคอมโพเนนต์ Navigator จะประกอบด้วยคอมโพเนนต์ Screen ของหน้าจอที่ต้องการกำหนดการทำ stack navigation

ตัวอย่างโปรแกรม App.js

```
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";

import Screen1 from "../screens/Screen1";
import Screen2 from "../screens/Screen2";
import Screen3 from "../screens/Screen3";

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="S1">
        <Stack.Screen name="S1" component={Screen1} />
        <Stack.Screen name="S2" component={Screen2} />
        <Stack.Screen name="S3" component={Screen3} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Route Names



การ Navigate ระหว่างหน้าจอ

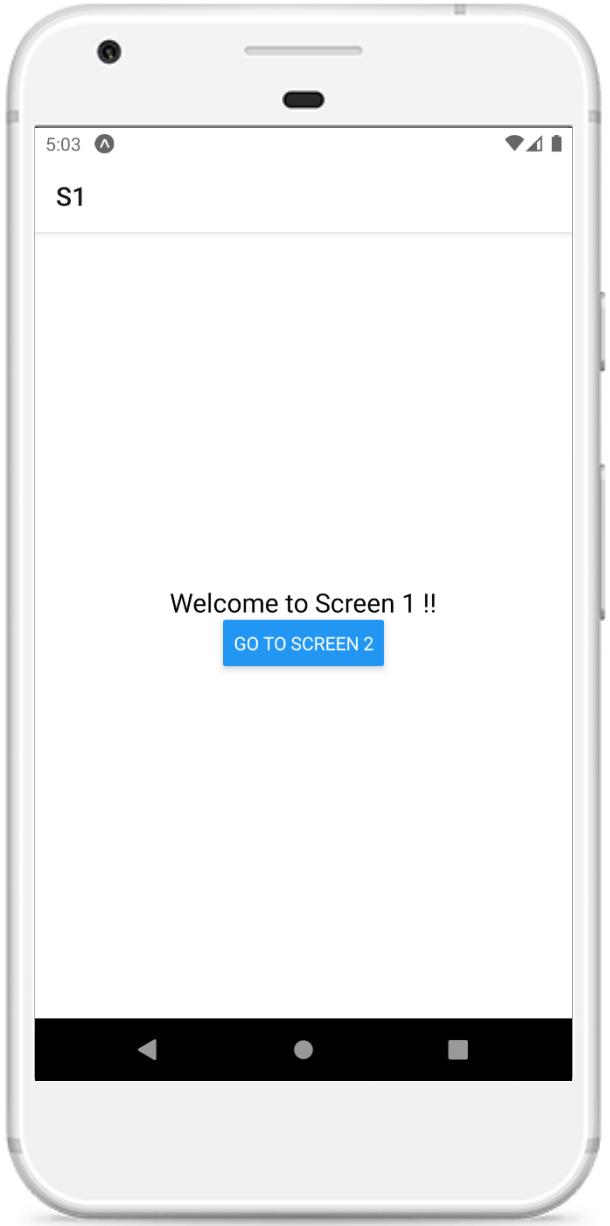
- ทุกคอมโพเนนต์ (หน้าจอ) ที่กำหนดใน navigation (ในที่นี้คือ Screen1, Screen2, Screen3) จะมี navigation props ที่สามารถจัดการการทำ navigation ได้
- เมธอด navigate: ใช้เปลี่ยนหน้าจอ
  - navigation.navigate(RouteName)
- กรณีที่ต้องการเปลี่ยนหน้าจอจาก Screen1 ไปยัง Screen2 สามารถเขียนโปรแกรมในฝั่ง Screen1 ดังนี้
  - navigation.navigate("S2")

เมธอดพื้นฐานเกี่ยวกับการทำ Navigation

- navigate : เปลี่ยนหน้าจอไป Route name ที่ต้องการ (แสดงหน้า Route name นั้น)
- push : เพิ่ม Route name ที่ต้องการลงบน stack (แสดงหน้า Route name นั้น)
- goBack : เปลี่ยนหน้าจอไปยังหน้าจอก่อนหน้า
- popToTop : เอา Route name ออกจาก stack ทั้งหมด เหลือเฉพาะ Route name แรกสุด (ย้อนไปหน้าจอแรกสุด)
- replace : เปลี่ยนหน้าจอไป Route name ที่ต้องการ แทนที่หน้าจอก่อนหน้า (ไม่สามารถย้อนกลับไปหน้าจอก่อนหน้าได้)

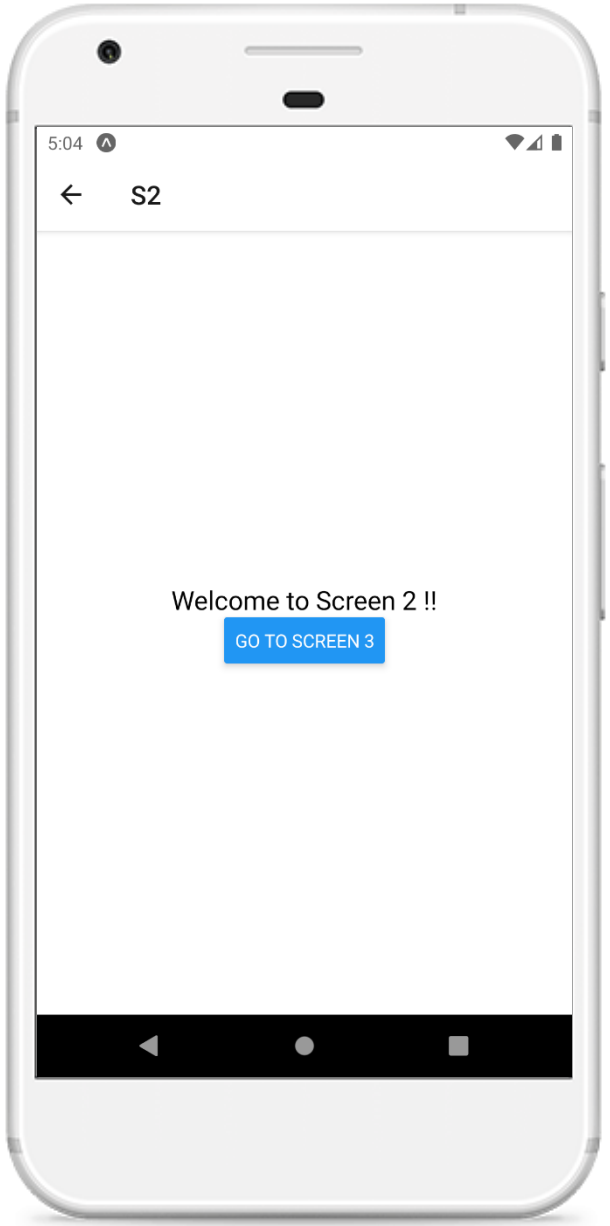
ตัวอย่างโปรแกรม Screen1.js

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

const Screen1 = ({navigation}) => {
  return (
    <View>
      <Text>Welcome to Screen 1 !!</Text>
      <Button
        title="Go to Screen 2"
        onPress={ () => { navigation.navigate("S2"); } }
      />
    </View>
  );
};
export default Screen1;
```



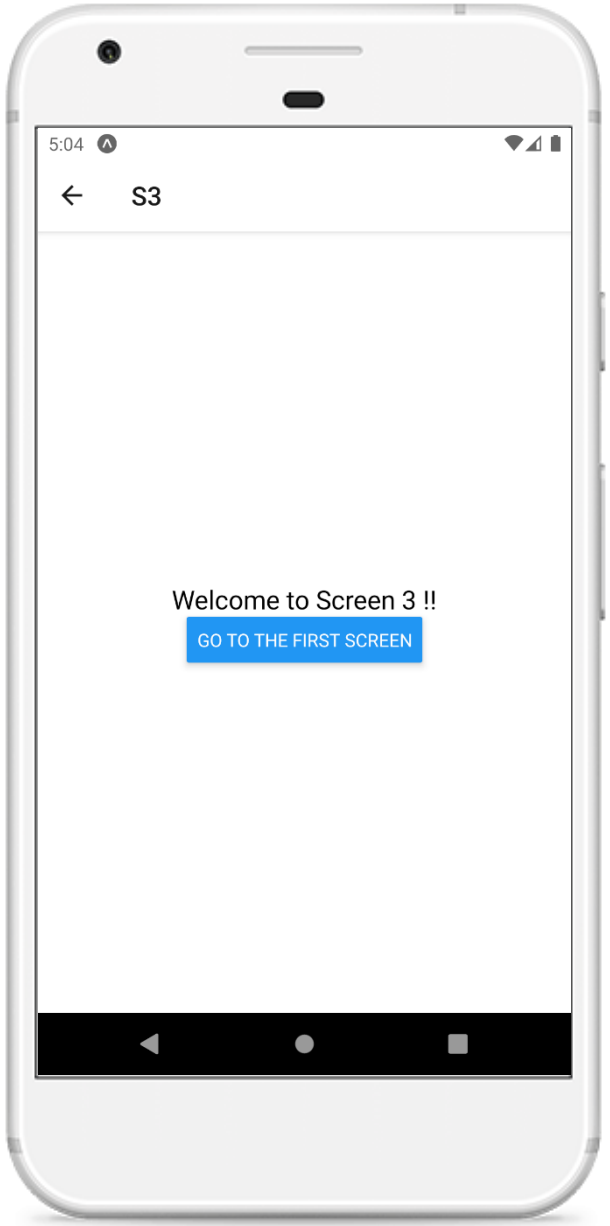
ตัวอย่างโปรแกรม Screen2.js

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

const Screen2 = ({navigation}) => {
  return (
    <View>
      <Text>Welcome to Screen 2 !!</Text>
      <Button
        title="Go to Screen 3"
        onPress={ () => { navigation.navigate("S3"); } }
      />
    </View>
  );
};
export default Screen2;
```



ตัวอย่างโปรแกรม Screen3.js

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

const Screen3 = ({navigation}) => {
  return (
    <View>
      <Text>Welcome to Screen 3 !!</Text>
      <Button
        title="Go to the first screen"
        onPress={ () => { navigation.popToTop(); } }
      />
    </View>
  );
};
export default Screen3;
```

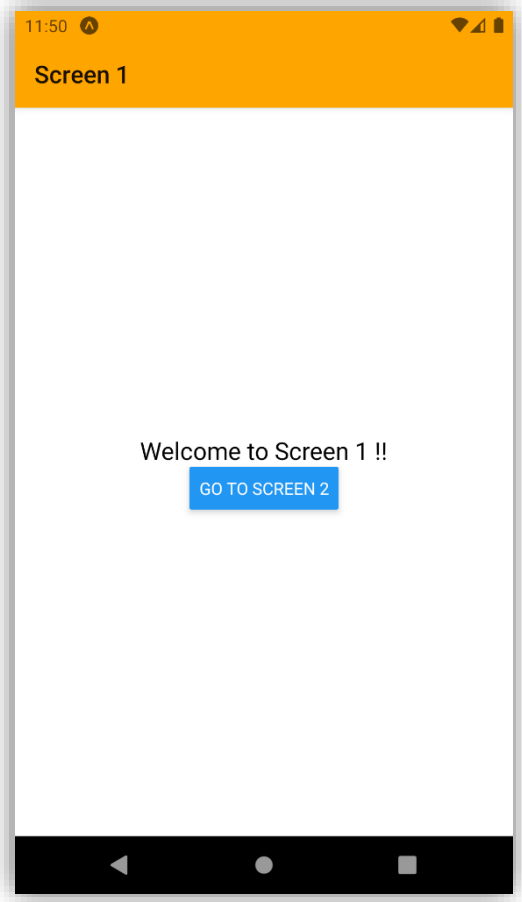


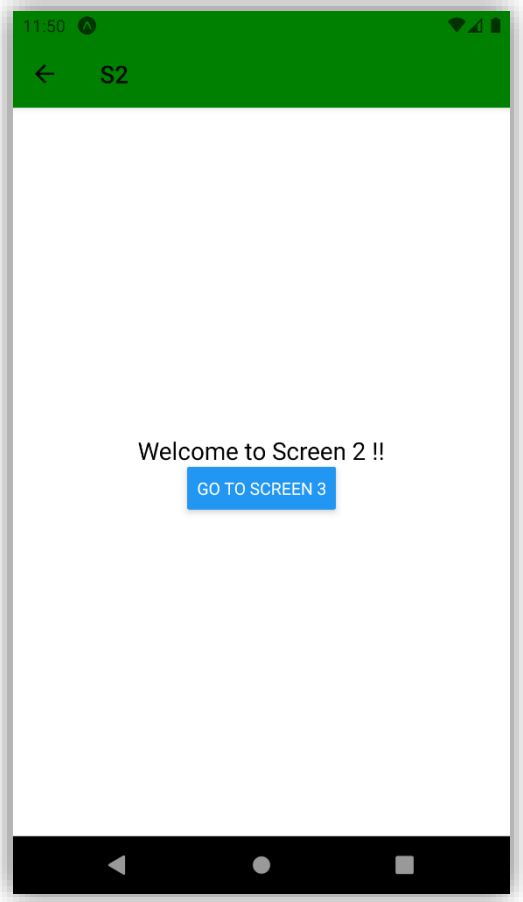
Options prop: ปรับแต่ง Header bar ของแต่ละ Screen

- React Navigation ได้มีการเตรียม options property เพื่อปรับแต่งการตั้งค่า Header bar ในแต่ละ Screen
- สามารถกำหนดค่า options ได้ในคอมโพเนนต์ Screen
- ตัวอย่าง key properties ที่ใช้การกำหนดค่า Header
  - title: ชื่อที่แสดงบนเฮดเดอร์
  - headerStyle: กำหนดสไตล์ของเฮดเดอร์ เช่น สีพื้นหลัง
  - headerTintColor: กำหนดสีปุ่ม Back หรือสีของชื่อเฮดเดอร์
  - headerTitleStyle: กำหนดสไตล์ตัวอักษรของชื่อเฮดเดอร์

ตัวอย่างโปรแกรม App.js (ใช้ options)

```
const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="S1">
        <Stack.Screen name="S1" component={Screen1}
            options={{
                title: "Screen1",
                headerStyle: { backgroundColor: "orange" },
            }} />
        <Stack.Screen name="S2" component={Screen2}
            options={{
                headerStyle: { backgroundColor: "green" },
            }} />
        <Stack.Screen name="S3" component={Screen3} />
      </Stack.Navigator>
    </NavigationContainer>
  ); }
```



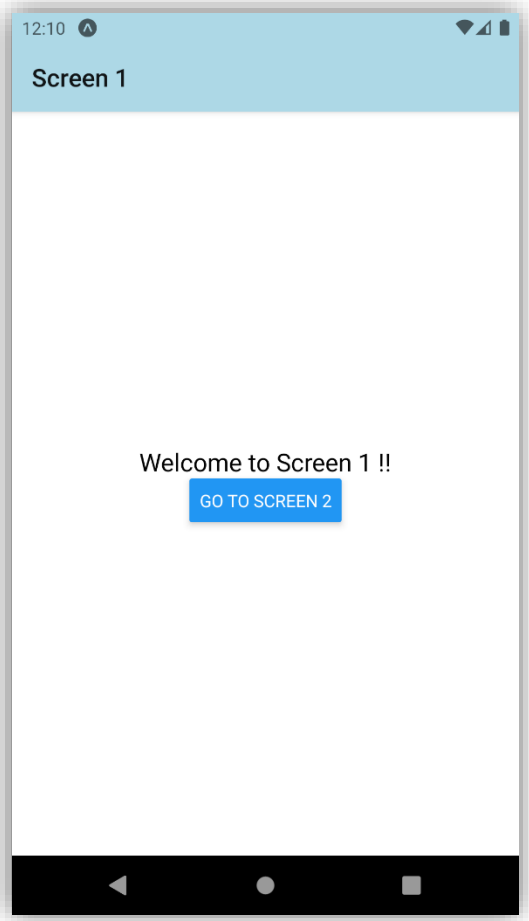


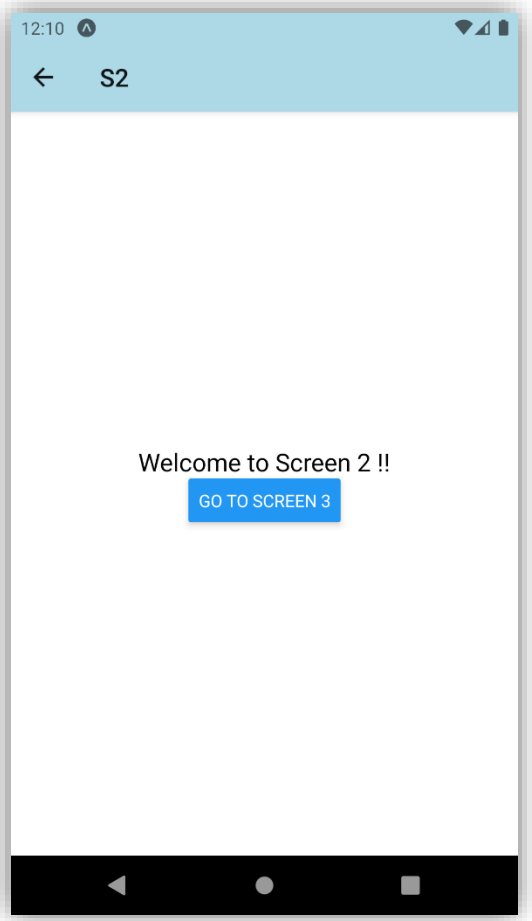
screenOptions

- กรณีที่ต้องการใช้ค่า options ร่วมกันในทุก Screen (ให้ทุกหน้าจอใน Navigator มีการกำหนดค่าเฮดเดอร์ในรูปแบบเดียวกัน) ทำได้โดยกำหนด screenOptions prop
  - กำหนด screenOptions ในคอมโพเนนต์ Stack Navigator
  - กำหนดค่าคล้ายกับการกำหนด options prop ในแต่ละ Screen

ตัวอย่างโปรแกรม App.js (ใช้ screenOptions)

```
const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName=“S1“
          screenOptions={{ headerStyle: { backgroundColor: "lightblue" } }}>
        <Stack.Screen name=“S1" component={Screen1}
            options={{ title: "Screen1“ }} />
        <Stack.Screen name=“S2" component={Screen2} />
        <Stack.Screen name=“S3" component={Screen3} />
      </Stack.Navigator>
    </NavigationContainer>
  ); }
```





การส่ง parameter ผ่าน routes

- เมื่อมีหน้าจอหลายหน้า ผลลัพธ์ของหน้าจอหนึ่งอาจขึ้นกับข้อมูลที่ได้รับมาจากหน้าจอก่อนหน้า ซึ่งส่งผ่านมาจาก params ใน route props ซึ่งถูกส่งมาในทุก Screen
  - เช่น กดเลือกคนในหน้า Contact โปรแกรมจะเปลี่ยนเป็นหน้า Profile แสดงรายละเอียดของคนๆ นั้น
- เราสามารถส่ง parameter ให้กับ Screen หน้าถัดไปได้ โดยมีรูปแบบ ดังนี้
  - navigation.navigate(RouteName, { /* params */ })
  - navigation.navigate(RouteName, {param1: val1, param2: val2, ...})
- ตัวอย่าง : navigation.navigate('S2', {prev: 'S1', id: 1})

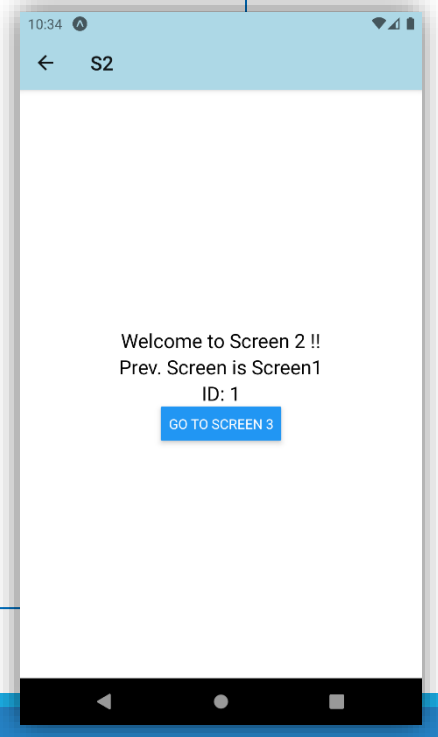
การรับ parameter ผ่าน params

- เมื่อมีการส่งข้อมูลมาจากหน้าจอก่อนหน้า เราสามารถรับข้อมูลได้ผ่าน route prop และเรียก params
  - route.params
- ตัวอย่าง :
  - const {prev, id} = route.params;

ตัวอย่างโปรแกรม (Screen1.js และ Screen2.js)

```
const Screen1 = ({navigation}) => {
  return (
    <View>
      <Text>Welcome to Screen 1 !!</Text>
      <Button
        title="Go to Screen 2"
        onPress={() => {
          navigation.navigate("S2", { prev: "Screen1", id: 1 });
        }}
      />
    </View>
  );
};
```

```
const Screen2 = ({route, navigation}) => {
  const {prev, id} = route.params;
return (
    <View style={styles.container}>
      <Text style={styles.content}>Welcome to Screen 2 !!</Text>
      <Text style={styles.content}>Prev. Screen is {prev}</Text>
      <Text style={styles.content}>ID: {id}</Text>
      <Button
        title="Go to Screen 3"
        onPress={() => {
          navigation.navigate("S3");
        }}
      />
    </View>
  );
};
```

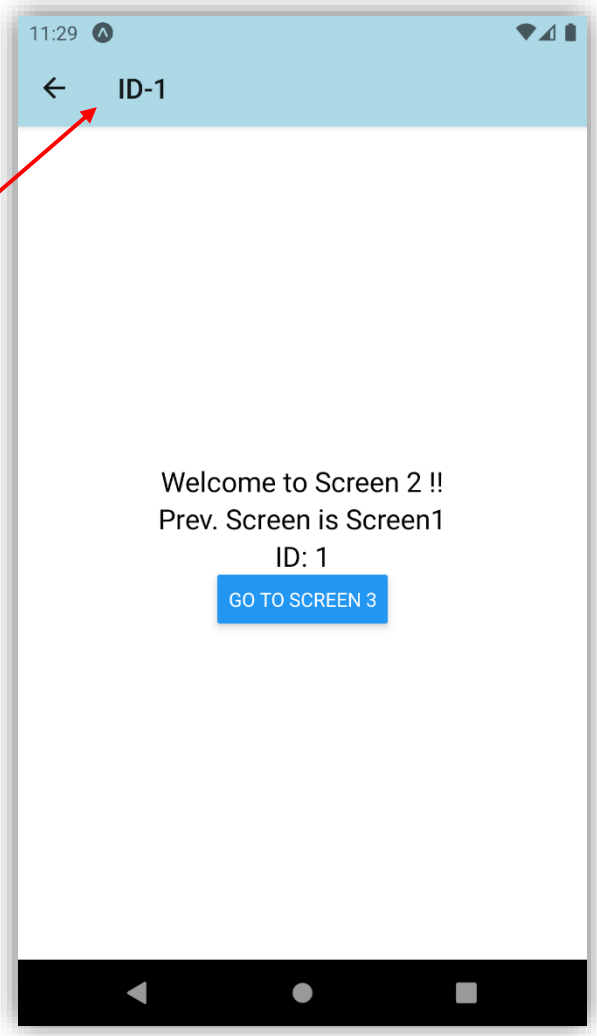


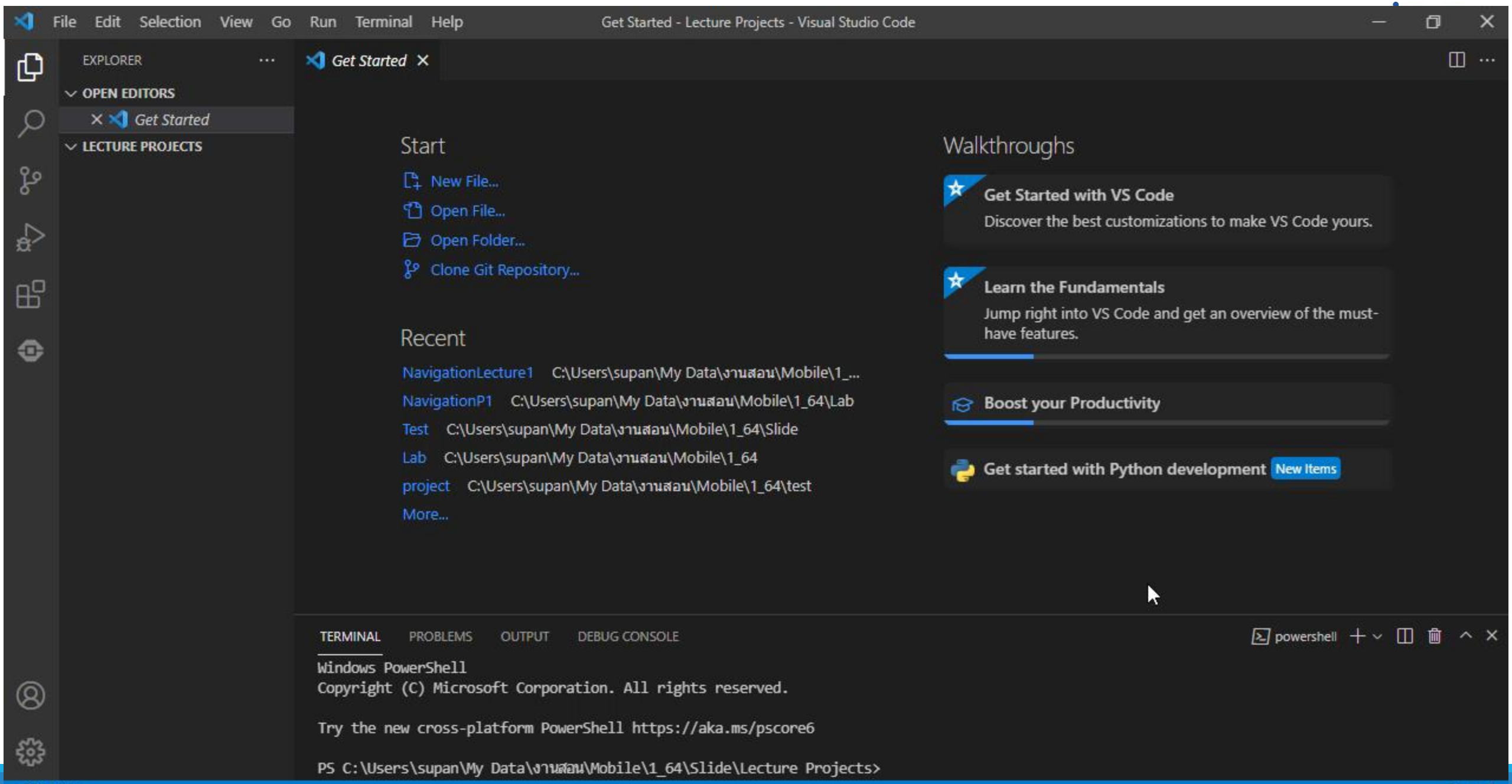
การปรับเฮดเดอร์ตามข้อมูลที่ได้รับจากหน้าจอก่อนหน้า

- นอกจากการกำหนด options เพื่อปรับค่าเฮดเดอร์ของแต่ละ Screen แบบกำหนดค่าคงที่แล้ว เรายังสามารถปรับคุณลักษณะของเฮดเดอร์ได้ตามข้อมูลที่รับมาจากหน้าจอก่อนหน้าได้
- สามารถกำหนดได้ใน options prop ใน Screen ของ Stack navigator ได้ตามปกติ
- สามารถรับค่า parameter ที่ส่งมาผ่าน route แล้วนำไปใช้ปรับแต่งเฮดเดอร์ได้
- จากตัวอย่างก่อนหน้า Screen1 มีการส่งค่าพารามิเตอร์ prev และ id ให้กับ Screen2 และ Screen2 สามารถกำหนด header title ให้เป็น id ที่รับเข้ามาได้
  - เขียนโปรแกรมจัดการในส่วนของ Stack Navigator ใน App.js

ตัวอย่างโปรแกรม App.js

```
const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="S1"
          screenOptions={{ headerStyle: { backgroundColor: "lightblue" } }}>
        <Stack.Screen name="S1" component={Screen1}  options={{ title: "Screen1" }} />
        <Stack.Screen name="S2" component={Screen2}
          options={
            ({ route }) => ({
              title: "ID-" + route.params.id.toString(),
            }) } />
        <Stack.Screen name="S3" component={Screen3} />
      </Stack.Navigator>
    </NavigationContainer>
  ); }
```




File Edit Selection View Go Run Terminal Help
Get Started - Lecture Projects - Visual Studio Code
EXPLORER
OPEN EDITORS
Get Started
LECTURE PROJECTS
Get Started
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Recent
NavigationLecture1 C:\Users\supan\My Data\งานสอน\Mobile\1_...
NavigationP1 C:\Users\supan\My Data\งานสอน\Mobile\1_64\Lab
Test C:\Users\supan\My Data\งานสอน\Mobile\1_64\Slide
Lab C:\Users\supan\My Data\งานสอน\Mobile\1_64
project C:\Users\supan\My Data\งานสอน\Mobile\1_64\test
More...
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Boost your Productivity
Get started with Python development
New Items
TERMINAL
PROBLEMS
OUTPUT
DEBUG CONSOLE
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\supan\My Data\งานสอน\Mobile\1_64\Slide\Lecture Projects>