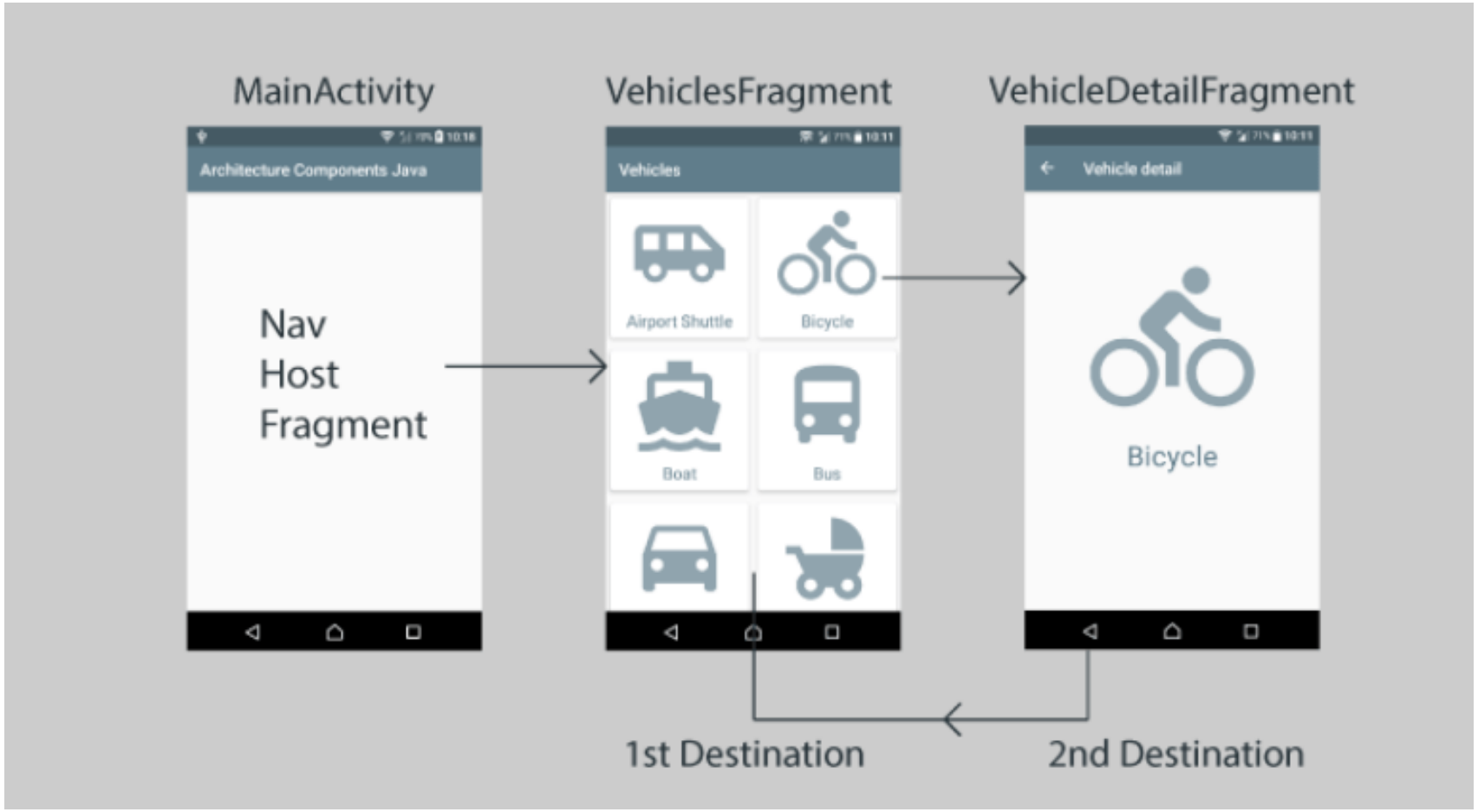
06016323 Mobile Device Programming

CHAPTER 9 : 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

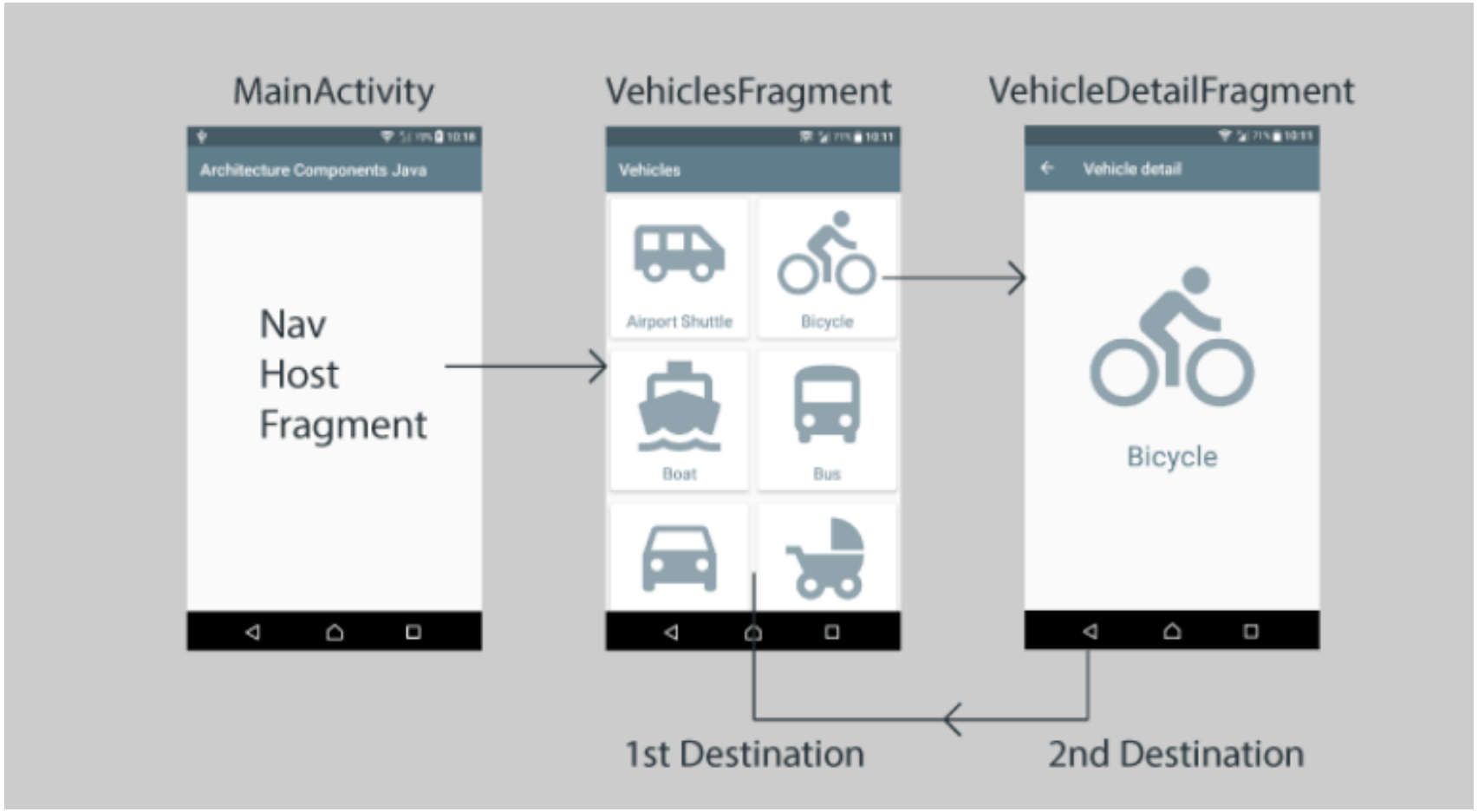
React Navigation

- Stack Navigation
- Tab Navigation
- Drawer Navigation

Stack Navigation

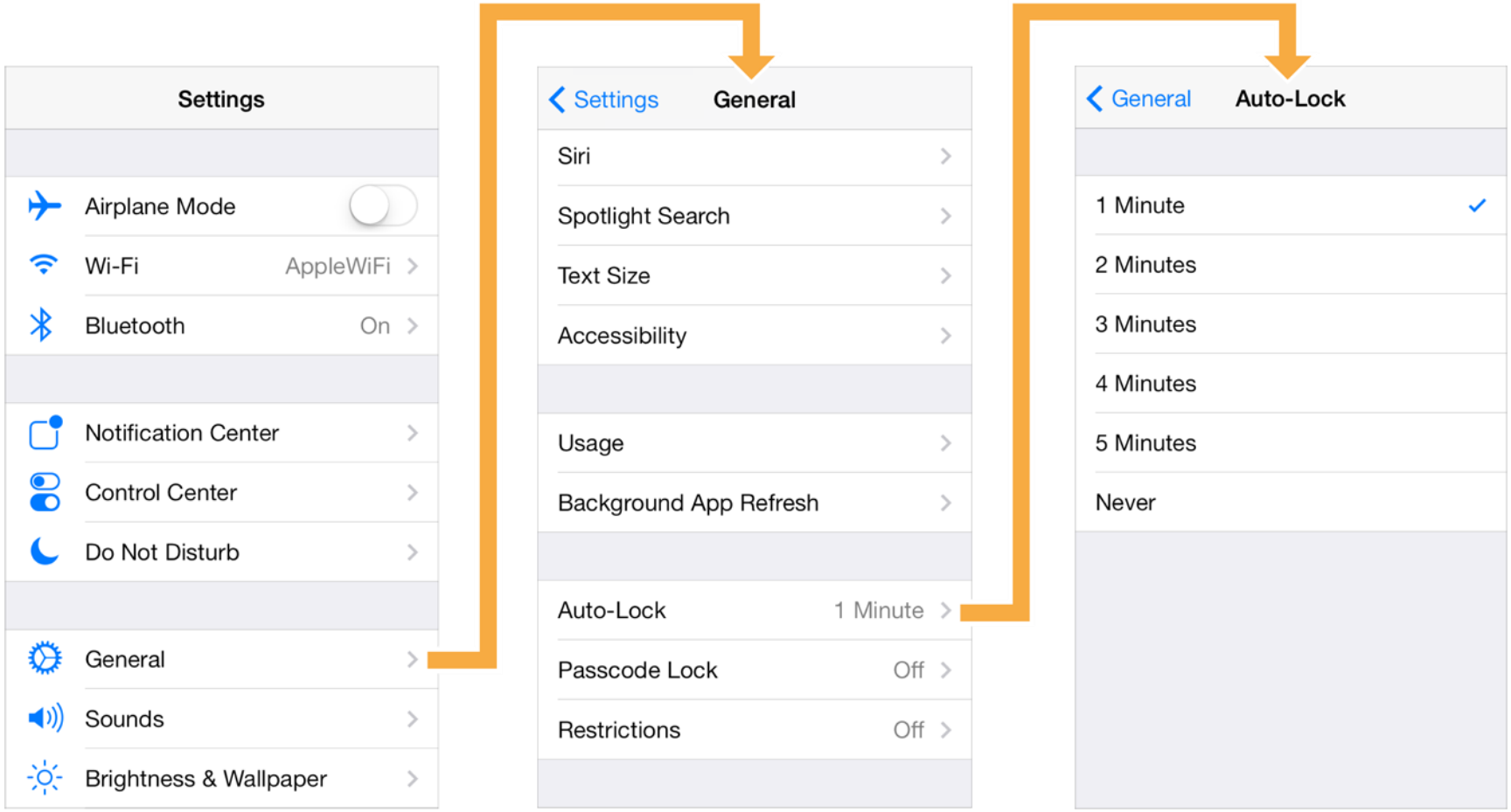
- การเปลี่ยนไปยังหน้าจออื่น Stack navigator จะทำการ push หน้าจอใหม่เข้าไปใน Navigation stack (อยู่ที่ส่วน top ของ stack)
- เมื่อมีการย้อนกลับไปหน้าจอก่อนหน้า Stack navigator จะทำการ pop หน้าจอล่าสุดออก
- หน้าจอที่ตำแหน่ง top ของ stack จะถูกแสดงให้ผู้ใช้เห็น ณ ขณะนั้น

Stack Navigation (Android)



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

Stack Navigation (iOS)



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

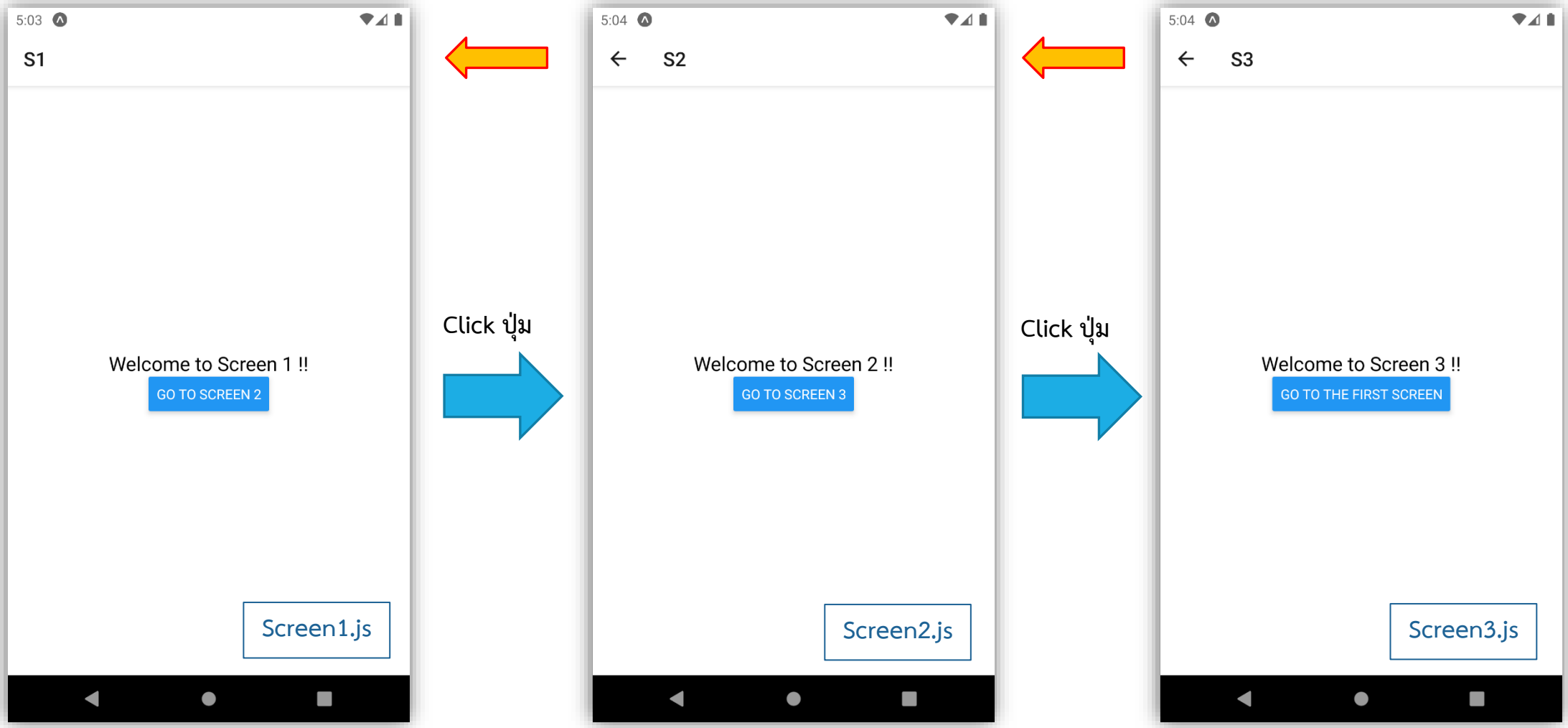
การเขียนโปรแกรมเพื่อจัดการ Navigation ใน React Native

- ติดตั้งไลบรารี react-navigation เพื่อใช้จัดการการทำ Navigation
  - npm install --save react-navigation หรือ
  - expo install react-navigation
- ติดตั้ง dependencies ที่ react-navigation ต้องการเพิ่มเติม
  - expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view

ติดตั้ง react-navigation-stack

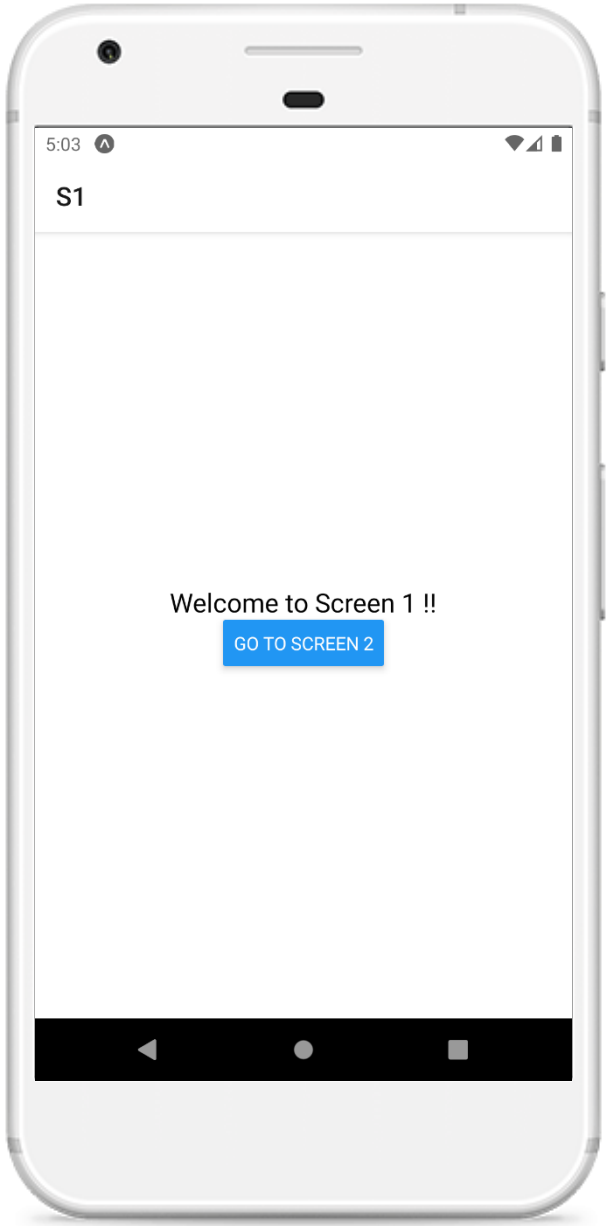
- สำหรับ react-navigation เวอร์ชั่น 4 ขึ้นไป ต้องมีการติดตั้งไลบรารีเพิ่มเติม เพื่อทำการสร้าง Navigator ในรูปแบบต่างๆ (ในที่นี้ จะสร้าง Stack Navigator)
  - npm install --save react-navigation-stack หรือ
  - expo install react-navigation-stack

ตัวอย่างแอปพลิเคชัน



ตัวอย่างโปรแกรม Screen1.js

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

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



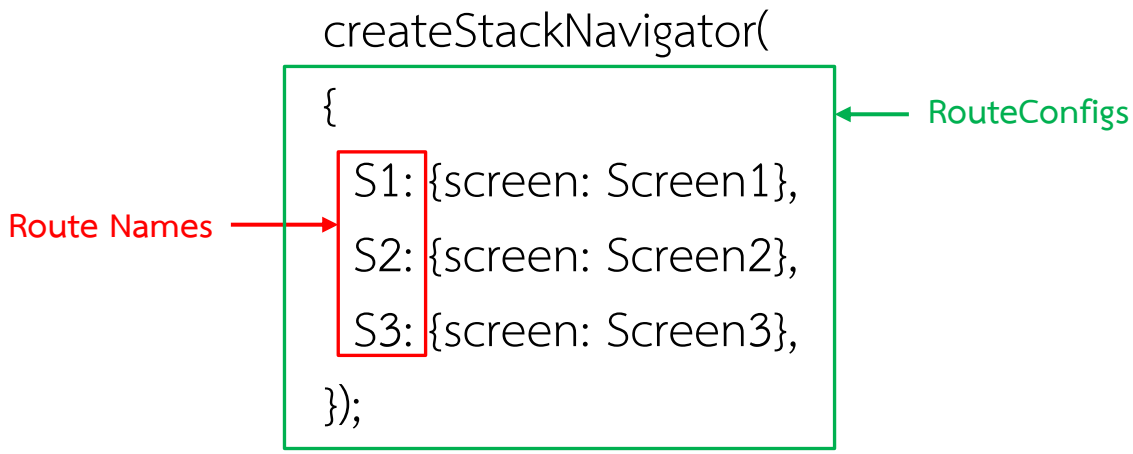
การสร้าง Stack Navigator

- หากต้องการทำ Stack Navigation ในแอปพลิเคชัน ต้องทำการสร้าง Navigator ที่ทำหน้าที่กำหนด navigation logic ขึ้นมาก่อน
  - import { createStackNavigator } from "react-navigation-stack"; // v.4 ขึ้นไป
  - import { createAppContainer } from "react-navigation";
  - สร้าง Stack navigator ด้วยเมธอด createStackNavigator()
  - เมื่อจะนำ Navigator ไปใช้ ต้อง export อยู่ในรูปแบบของ Navigation container ด้วยเมธอด createAppContainer()

createStackNavigator()

- รูปแบบ : createStackNavigator(RouteConfigs, StackNavigatorConfig);
  - RouteConfigs: การกำหนด configuration ให้กับ route ต่างๆ ใน navigation
  - StackNavigatorConfig: การกำหนด configuration ให้กับ stack navigator
  - อาจเขียนเฉพาะ RouteConfigs ได้
- ตัวอย่าง

```
createStackNavigator(
{
  S1: Screen1,
  S2: Screen2,
  S3: Screen3,
});
```



ตัวอย่างโปรแกรม MyNavigator.js

```
import { createStackNavigator } from "react-navigation-stack";
import { createAppContainer } from "react-navigation";

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

const MyNavigator = createStackNavigator({
  S1: Screen1,
  S2: Screen2,
  S3: Screen3,
});

export default createAppContainer(MyNavigator);
```

การ Navigate ระหว่างหน้าจอ

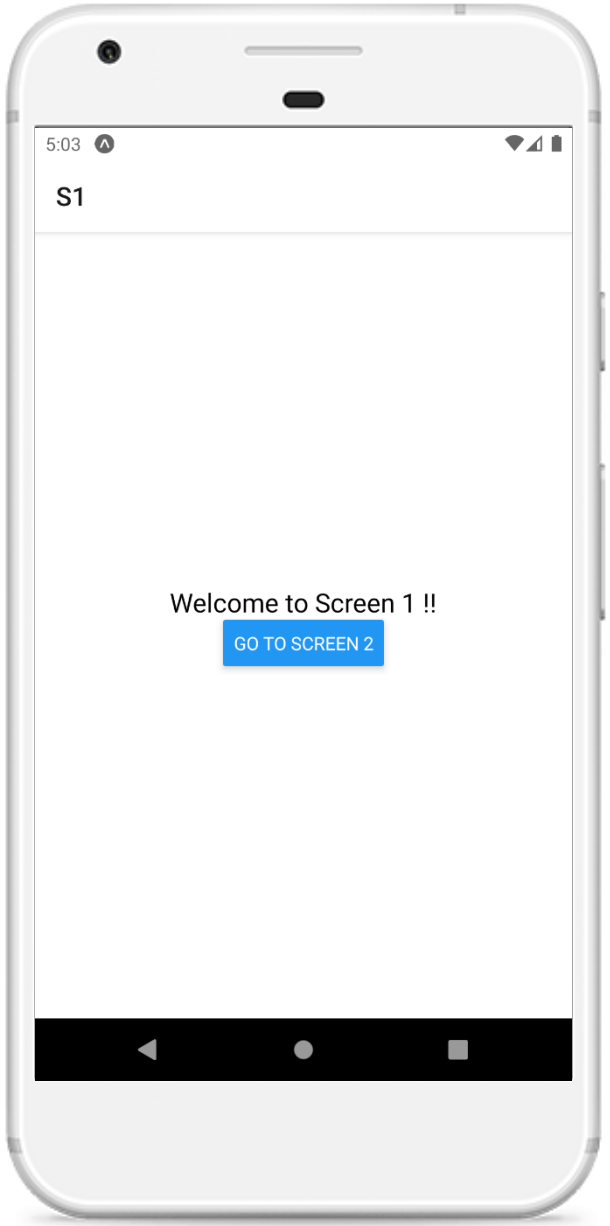
- ทุกคอมโพเนนต์ (หน้าจอ) ที่กำหนดใน navigation (ในที่นี้คือ Screen1, Screen2, Screen3) จะมี navigation props ที่สามารถจัดการการทำ navigation ได้
- เมธอด navigate: ใช้เปลี่ยนหน้าจอ
  - props.navigation.navigate(RouteName)
- กรณีที่ต้องการเปลี่ยนหน้าจอจาก Screen1 ไปยัง Screen2 สามารถเขียนโปรแกรมในฝั่ง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

เมธอดพื้นฐานเกี่ยวกับการทำ Navigation

- navigate : เปลี่ยนหน้าจอไป Route name ที่ต้องการ (แสดงหน้า Route name นั้น)
- push : เพิ่ม Route name ที่ต้องการลงบน stack (แสดงหน้า Route name นั้น)
- pop : เอา Route name ที่ตำแหน่ง Top ของ stack ออก (ย้อนไปหน้าจอก่อนหน้า)
- popToTop : เอา Route name ออกจาก stack ทั้งหมด เหลือเฉพาะ Route name แรกสุด (ย้อนไปหน้าจอแรกสุด)
- replace : เปลี่ยนหน้าจอไป Route name ที่ต้องการ แทนที่หน้าจอก่อนหน้า (ไม่สามารถย้อนกลับไปหน้าจอก่อนหน้าได้)

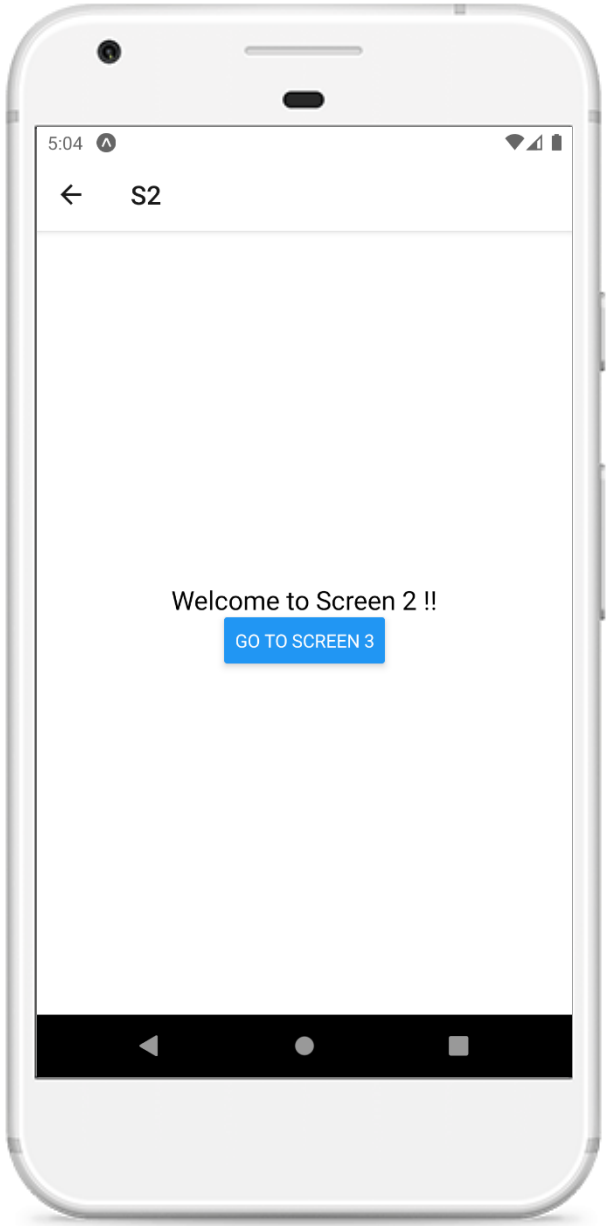
ตัวอย่างโปรแกรม Screen1.js

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

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



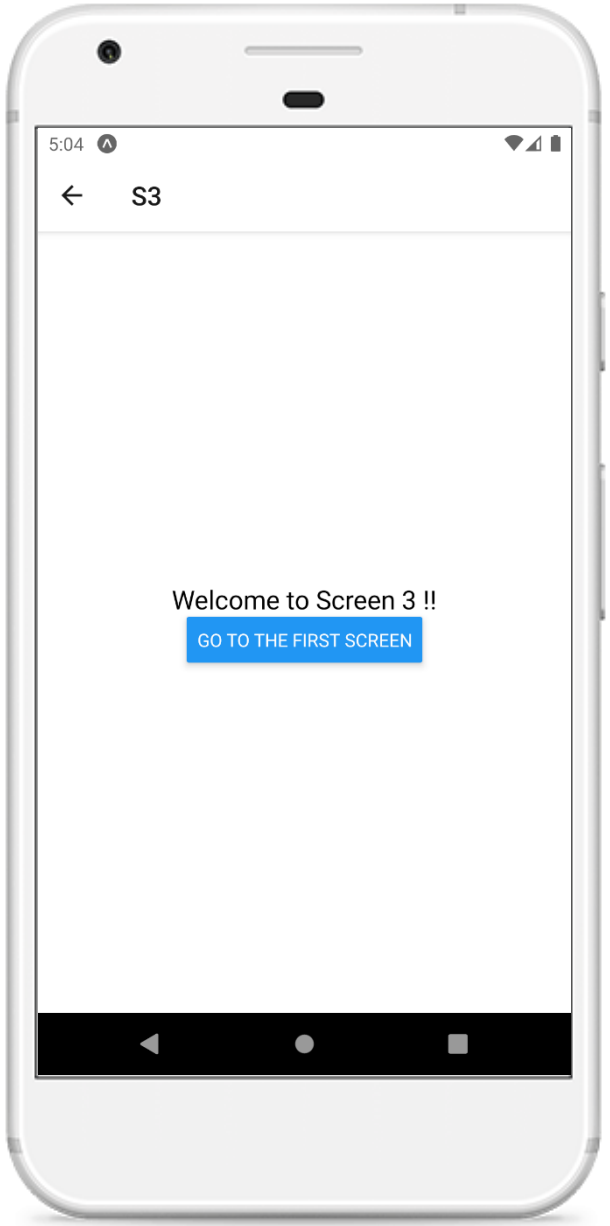
ตัวอย่างโปรแกรม Screen2.js

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

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



ตัวอย่างโปรแกรม Screen3.js

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

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

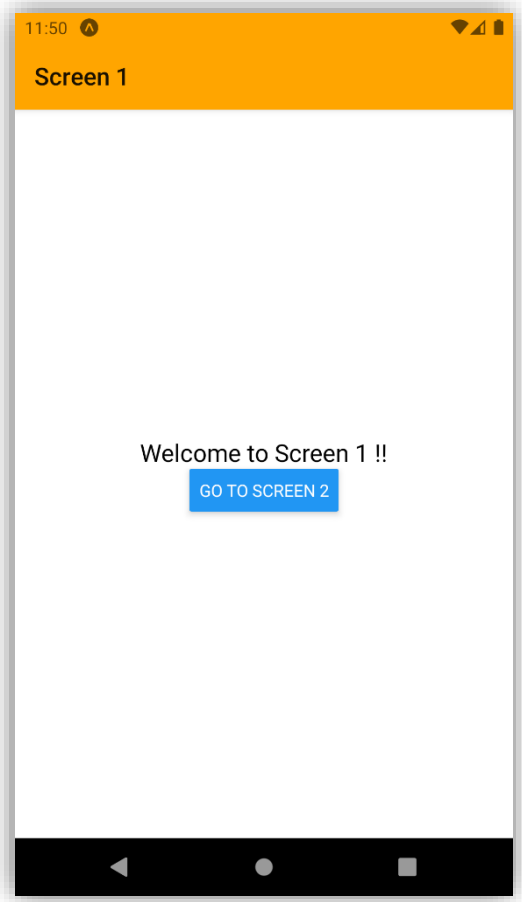


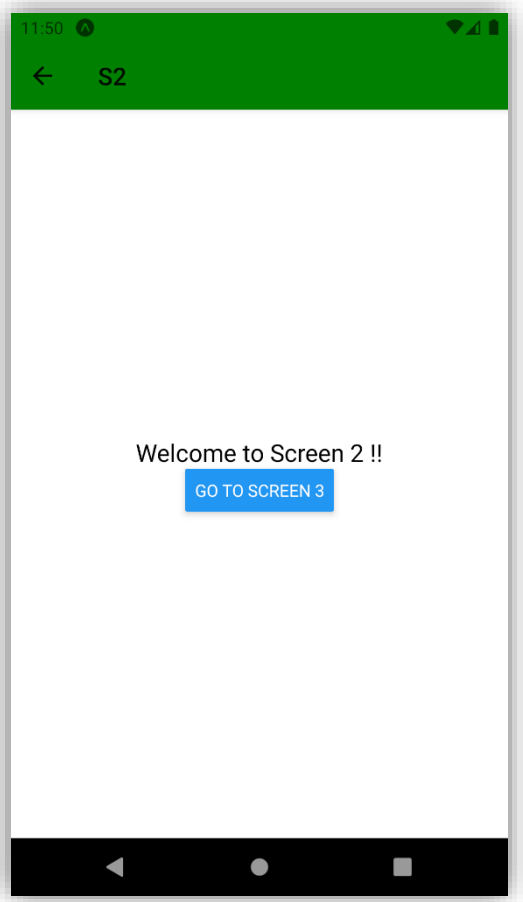
Navigation Screen Options (navigationOptions)

- React Navigation ได้มีการเตรียม navigationOptions property เพื่อปรับแต่งการตั้งค่า navigation ในแต่ละหน้าจอ
- สามารถกำหนดค่า navigationOptions ได้ใน createStackNavigator ได้

ตัวอย่างโปรแกรม MyNavigator.js (ใช้ navigationOptions)

```
const MyNavigator = createStackNavigator({
  S1: {
    screen: Screen1,
    navigationOptions: {
      title: "Screen 1",
      headerStyle: { backgroundColor: "orange" },
    },
  },
  S2: {
    screen: Screen2,
    navigationOptions: {
      headerStyle: { backgroundColor: "green" },
    },
  },
  S3: { screen: Screen3 },
});
```





defaultNavigationOptions

- กรณีที่ต้องการปรับแต่งค่า navigation ให้เหมือนกันทุกหน้าจอ
  - createStackNavigator(RouteConfigs, StackNavigatorConfig);
  - สามารถกำหนดด้วย StackNavigatorConfig
  - ใช้ property defaultNavigationOptions

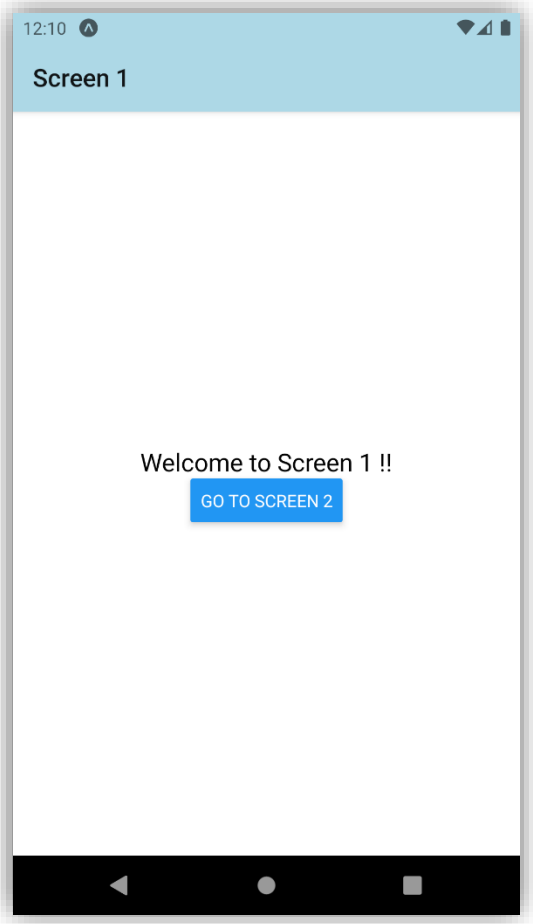
ตัวอย่างโปรแกรม MyNavigator.js (ใช้ defaultNavigationOptions)

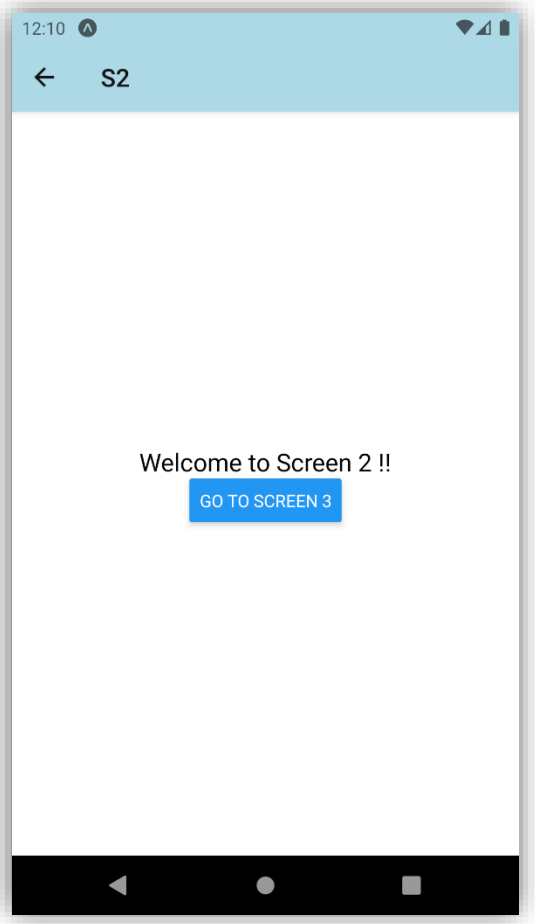
```
const MyNavigator = createStackNavigator(
{
  S1: {
    screen: Screen1,
    navigationOptions: {
      title: "Screen 1",
    },
  },
  S2: { screen: Screen2 },
  S3: { screen: Screen3 },
},
```

RouteConfigs

StackNavigatorConfigs

```
{
  defaultNavigationOptions: {
    headerStyle: { backgroundColor: "lightblue" },
  },
}
);
```





การส่งข้อมูลผ่าน Navigation parameters

- เมื่อมีหน้าจอหลายหน้า ผลลัพธ์ของหน้าจอหนึ่งอาจขึ้นกับข้อมูลที่ได้รับมาจากหน้าจอก่อนหน้า ซึ่งส่งผ่านมาจาก navigation parameters ในเมธอด navigate
  - เช่น กดเลือกคนในหน้า Contact โปรแกรมจะเปลี่ยนเป็นหน้า Profile แสดงรายละเอียดของคนๆ นั้น
- รูปแบบ :
  - navigate(RouteName, NavParams)
  - navigate(RouteName, {param1: val1, param2: val2, ...})
- ตัวอย่าง : navigate('S2', {prev: 'S1', id: 1})

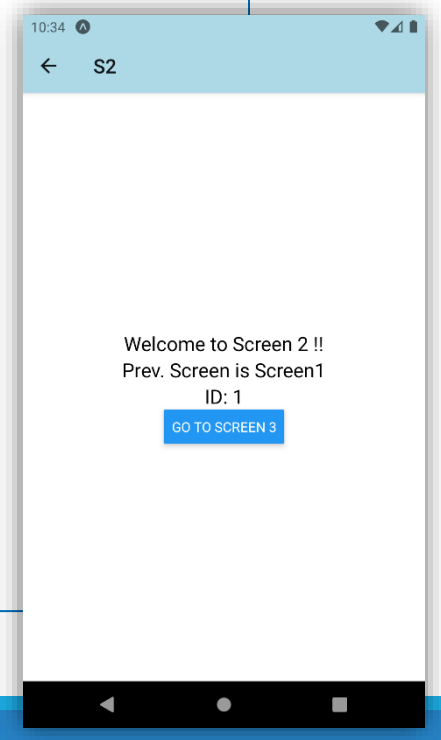
การรับข้อมูลผ่าน Navigation parameters

- เมื่อมีการส่งข้อมูลมาจากหน้าจอก่อนหน้า เราสามารถรับข้อมูลได้ผ่าน Navigation property (props) และเมธอด getParam()
  - props.navigation.getParam(paramName)
- ตัวอย่าง :
  - const prev = props.navigation.getParam('prev’);
  - const id = props.navigation.getParam('id');

ตัวอย่างโปรแกรม

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

```
const Screen2 = (props) => {
  const prev = props.navigation.getParam("prev");
  const id = props.navigation.getParam("id");
  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={() => {
          props.navigation.navigate("S3");
        }}
      />
    </View>
  );
};
```



Dynamic Navigation parameters

- นอกจากการกำหนด navigationOptions จากการ createStackNavigator() แล้ว เรายังกำหนดค่า navigationOptions แยกในไฟล์คอมโพเนนต์นั้นๆ ได้อีกด้วย
  - ช่วยทำให้เราปรับคุณลักษณะของ header ได้ตามข้อมูลที่รับมาจากหน้าจอก่อนหน้าได้
- เรียกเมธอด navigationOptions โดยมีอาร์กิวเมนต์ navigationData
- ใช้ navigationData ทำการ getParam() จากหน้าจอก่อนหน้า เพื่ออัพเดท Header ได้
  - navigationData.navigation.getParam(paramName)

ตัวอย่างโปรแกรม

```
const Screen2 = (props) => {
  const prev = props.navigation.getParam("prev");
  const id = props.navigation.getParam("id");
  return (
    <View style={styles.container}>
      <Text style={styles.content}>Welcome to Screen 2 !!</Text>
      ...
      ...
    </View>
  );
};

Screen2.navigationOptions = (navigationData) => {
  const id = navigationData.navigation.getParam("id");
  const newHead = "ID-" + id.toString();
  return { title: newHead };
};
```