

3SA04 – React Native

1. เครื่องมือที่จำเป็น

- Chocolatey (for Windows), Brew (for OSX)
- Node.js
- Yarn
- Git
- expo CLI
- Visual Studio Code
- Android Studio

หลังจากที่ได้ติดตั้ง Chocolatey ในเครื่องแล้ว สามารถติดตั้ง Node.js, Yarn และ Git ได้ผ่าน Chocolatey ผ่าน Command Prompt (ที่รันด้วยสิทธิ์ Administrator)

```
>> choco install nodejs
>> choco install yarn
>> choco install git
```

ในการติดตั้ง create-react-app CLI สามารถทำได้ผ่านการใช้คำสั่ง yarn ผ่าน Command Prompt (ที่รันด้วยสิทธิ์ Administrator)

```
>> yarn global add expo-cli
```

** สำหรับคอมพิวเตอร์ให้ห้องปฏิบัติการ เครื่องมือข้างต้นได้ติดตั้งไว้ล่วงหน้าเรียบร้อยแล้ว

ในการทดลองนี้ นศ. จะต้องติดตั้งโปรแกรม Expo (มีทั้งบน Android และ iOS) ลงบนสมาร์ตโฟนที่ใช้ในการรันโปรแกรม

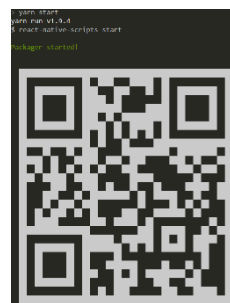
2. Hello world

สร้างโครงร่างโปรเจกต์สำหรับการพัฒนา React Native ด้วย expo (ให้เลือก template เป็น blank)

```
>> expo init wt-app
>> cd wt-app
```

ทำการรันโปรแกรมโปรแกรมขึ้นมา แล้วใช้สมาร์ตโฟนรันโปรแกรม expo แล้วสแกน QR Code ที่ได้จากคำสั่ง yarn start

```
>> yarn start
```



Open up App.js to start working on your app!
Changes you make will automatically reload.
Shake your phone to open the developer menu.



Source Code

เปิด Source Code ของโปรเจ็ค wt-app ด้วย Visual Studio Code หรือ Text Editor ที่ต้องการ แก้ไข App.js เหมือนโค้ดข้างล่าง

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  const doIt = () => {
    console.log("Hello from console")
  }
  return (
    <View style={styles.container}>
      <Text onPress={doIt}>Hello world</Text>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

สังเกตผลลัพธ์ที่ได้บนโปรแกรม Expo ในสมาร์ตโฟน

Passing Props

สร้างโพลเตอร์ components ในโปรเจ็ค แล้วสร้างไฟล์ Weather.js

คอมโพเนนต์ Weather รับ Props ชื่อ zipCode โดยให้กำหนด zipCode เป็น 90110

ไฟล์ Weather.js

```
export default function Weather(props) {
  return (
    <Text>{props.zipCode}</Text>
  );
}
```

ไฟล์ App.js

```
<View style={styles.container}>
  <Weather zipCode="90110"/>
  <StatusBar style="auto" />
</View>
```

Components and Image Background

กำหนด State ให้กับคอมโพเนนต์ใน constructor แล้วใช้เป็น props ส่งผ่านไปยังคอมโพเนนต์ Forecast ที่สร้างขึ้นใหม่

ไฟล์ Forecast.js

```
export default function Forecast(props) {
  return (
    <View >
      <Text>{props.main}</Text>
      <Text>{props.description}</Text>
      <View>
        <Text>{props.temp}</Text>
        <Text>°C</Text>
      </View>
    </View>
  );
}
```

ไฟล์ Weather.js และรูป background ที่เหมาะสม

```
export default function Weather(props) {
  const [forecastInfo, setForecastInfo] = useState({
    main: '-',
    description: '-',
    temp: 0
  })
  return (
```

```

    <View>
      <ImageBackground source={require('../bg.jpg')} style={styles.backdrop}>
        <Text>Zip Code</Text>
        <Text>{props.zipCode}</Text>
        <Forecast {...forecastInfo} />
      </ImageBackground>
    </View>
  );
}

const styles = StyleSheet.create({
  backdrop: {
    alignItems: 'center',
    width: '100%',
    height: '100%'
  },
});





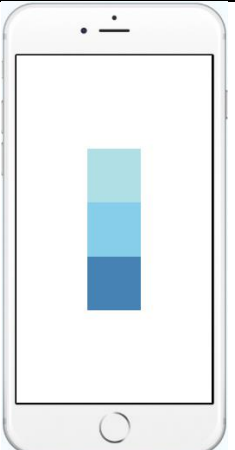
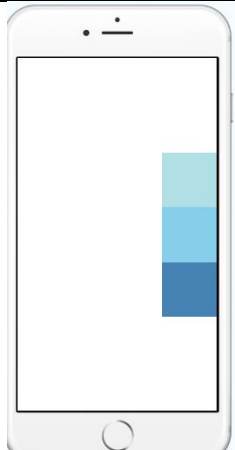
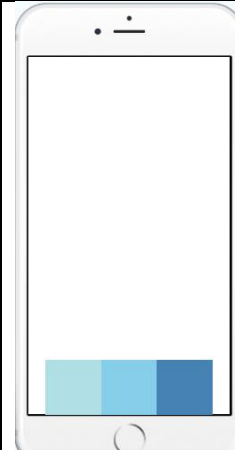
```

3. Flex Box

การจัด Layout บน React Native จะใช้ Flex Box ในการจัดการ แม้ว่า เทคนิคการจัด Flex Box จะรองรับความซับซ้อนสูง (React Native ไม่สนับสนุนทุก features ของ Flex Box) อย่างไรก็ตามคุณสมบัติที่ถูกใช้บ่อยในการจัด Layout มี 3 คุณสมบัติ คือ

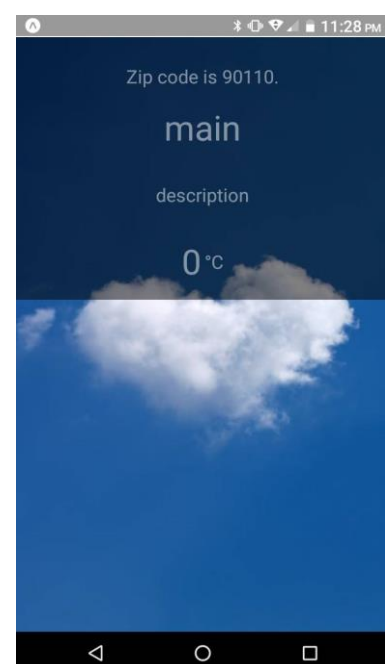
- flex – เป็นตัวเลข น้ำหนักในการแบ่งพื้นที่ เช่นถ้าคอมโพเนนต์ A มี flex เท่ากับ 1, คอมโพเนนต์ B มี flex เท่ากับ 2 หมายความว่า B จะใช้พื้นที่มากกว่า A สองเท่า และถ้ามีคอมโพเนนต์ A เพียง คอมโพเนนต์เดียว จะใช้เต็มพื้นที่
- flexDirection – แกนหลักของ Layout ว่าคอมโพเนนต์ลูกควรจัดเรียงแนวนอน (row) และแนวตั้ง (column) โดยค่า default คือ column
- justifyContent – การกระจายตัวของคอมโพเนนต์ลูกว่าควรจะเป็นแบบใด ในแนวแกนเดียวกับ flexDirection
- alignItems – การกระจายตัวของคอมโพเนนต์ลูกว่าควรจะเป็นแบบใด ในแนวแกนกับ flexDirection

ด.ย.

flexDirection: 'row'	flexDirection: 'row'	flexDirection: 'row'	flexDirection: 'row'
justifyContent: 'space-between'	justifyContent: 'center'	justifyContent: 'space-evenly'	justifyContent: 'flex-end'
			
flexDirection: 'column'	flexDirection: 'column'	flexDirection: 'row',	flexDirection: 'column'
justifyContent: 'center'	justifyContent: 'center'	justifyContent: 'center'	justifyContent: 'flex-end'
alignItems: 'center'	alignItems: 'flex-end'	alignItems: 'flex-end'	alignItems: 'flex-start'
			

CHALLENGE

จัด Layout โดยใช้ Flex Box และปรับสไตล์ของตัวอักษรและ background
เพิ่มเติม โดยใช้ height, paddingRight, backgroundColor, fontSize, color,
textAlign, textAlignVertical



4. Connect

เพิ่ม useEffect ลงไปในคอมโพเนนต์ Weather (ทำการ sign up แบบฟรีที่

https://home.openweathermap.org/users/sign_up เพื่อรับ APPID)

```
useEffect(() => {
  console.log(`fetching data with zipCode = ${props.zipCode}`)
  if (props.zipCode) {
    fetch(`http://api.openweathermap.org/data/2.5/weather?q=${props.zipCode},th&units=metric&APPID=...`)
      .then((response) => response.json())
      .then((json) => {
        setForecastInfo({
          main: json.weather[0].main,
          description: json.weather[0].description,
          temp: json.main.temp
        });
      })
      .catch((error) => {
        console.warn(error);
      });
  }
}, [props.zipCode])
```

5. Router

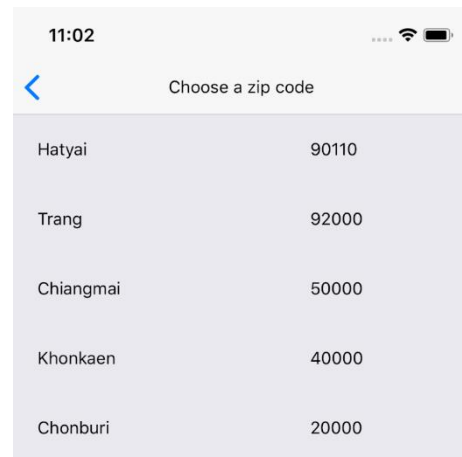
Application ที่สมบูรณ์มักมีหน้าจอ UI (สกรีน) มากกว่า 1 หน้าจอ ในการสลับหน้าจอไปมา สามารถทำได้ผ่านการใช้ Navigation Library ซึ่งไลบรารีที่ถือเป็น Official จาก React คือ react-navigation

```
>> expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view
>> yarn add @react-navigation/stack @react-navigation/native
```

เพิ่มคอมโพเนนต์ ZipCodeScreen ซึ่งเป็นหน้าจอสำหรับเลือกรหัสไปรษณีย์ (zip code) จากรายการที่กำหนดไว้ ทั้งนี้ นักศึกษาจะต้องกำหนด Style ให้เหมาะสมด้วยตนเอง

```
const availableZipItems = [
  { place: 'Hatyai', code: '90110' },
  { place: 'Trang', code: '92000' },
  { place: 'Chiangmai', code: '50000' },
  { place: 'Khonkaen', code: '40000' },
  { place: 'Chonburi', code: '20000' },
]

const ZipItem = ({place, code, navigation}) => (
  <View>
    <Text>{place}</Text>
    <Text>{code}</Text>
  </View>
)
```



```
const _keyExtractor = item => item.code

export default function ZipCodeScreen(){
  const navigation = useNavigation()
  return (
    <View>
      <FlatList
        data={availableZipItems}
        keyExtractor={_keyExtractor}
        renderItem={({item}) => <ZipItem {...item} navigation={navigation}/>}
      />
      <StatusBar style="auto" />
    </View>
  );
}
```

ปรับ App.js ให้ render ผลลัพธ์จากไลบรารี react-navigation แทนการ render คอมโพเนนต์ Weather โดยตรง

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

const Stack = createStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={ZipCodeScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Navigation & Route Parameter

ในการทำงานร่วมกันของแต่ละหน้าจอ UI เราสามารถส่งผ่านค่าการทำงานได้โดยใช้ Route Parameter

เพิ่มคอมโพเนนต์ WeatherScreen สำหรับหน้าจอแสดงคอมโพเนนต์ Weather

```
export default function WeatherScreen({route}) {
  return (
    <View>
      <Weather zipCode={route.params.zipCode} />
      <StatusBar style="auto" />
    </View>
  );
}
```

เพิ่ม Screen ลงใน App.js

```
<Stack.Navigator>
  <Stack.Screen name="Home" component={ZipCodeScreen} />
  <Stack.Screen name="Weather" component={WeatherScreen} />
</Stack.Navigator>
```

ปรับ ZipItem ในไฟล์ ZipCodeScreen.js เพื่อสร้างลิงค์ไปยังหน้า Weather พร้อมส่งผ่าน Route Parameter

```
const ZipItem = ({place, code, navigation}) => (
  <TouchableHighlight onPress={() => navigation.navigate('Weather', { zipCode: code})}>
    <View>
      <Text>{place}</Text>
      <Text>{code}</Text>
    </View>
  </TouchableHighlight>
)
```

6. งานหลังการทดลอง

ศึกษาข้อมูลเพิ่มเติมเกี่ยวกับ React Native แล้วให้ นศ. ปรับปรุงการทำงานของ Weather App ตามที่นศ. ต้องการ แล้วทำการ push ไว้ใน Github โดยจะต้องมีการ commit และ push ระหว่างทำอย่างน้อย 5 commit

ให้เขียนสรุปสิ่งที่ทำไว้ (เป็น text file หรือ md format) ที่ root ของ project โดยให้ตั้งชื่อว่า submission.txt หรือ submission.md แล้ว push มาด้วยใน Github ในไฟล์จะต้องระบุ ชื่อ-สกุล และรหัสนักศึกษาไว้ด้วย ทั้งนี้ให้ นศ. ทำการจับภาพหน้าจอ บันทึกไฟล์ (สามารถตั้งชื่อได้เอง) ไว้ใน root ของ project แล้ว push มาด้วย โดยหากมีการแก้ไขมากกว่าหนึ่งหน้า ก็ให้ส่งภาพมาทุกหน้า