3SA04 – React Native

# 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

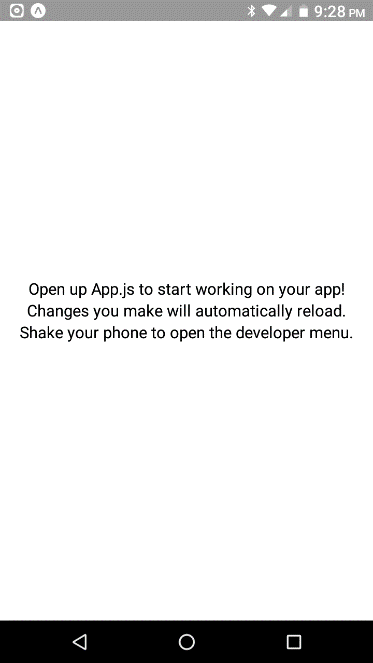
ในการติดตั้ง create-react-app CLI สามารถทำผ่านการใช้คำสั่ง yarn ผ่าน Command Prompt (ที่รันด้วยสิทธิ์ Administrator)

>> yarn global add expo-cli

\*\* สำหรับคอมพิวเตอร์ให้ห้องปฏิบัติการ เครื่องมือข้างต้นได้ติดตั้งไว้ล่วงหน้าเรียบร้อยแล้ว

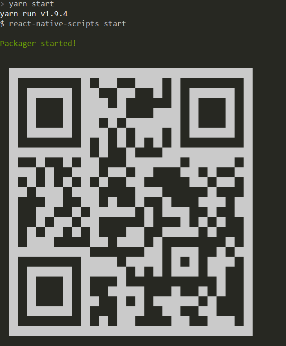
ในการทดลองนี้ นศ. จะต้องติดตั้งโปรแกรม Expo (มีทั้งบน Android และ iOS) ลงบนสมาร์ทโฟนที่ใช้ในการรันโปรแกรม

# Hello world

สร้างโครงร่างโปรเจ็คสำหรับการพัฒนา React Native ด้วย expo (ให้เลือก template เป็น blank)

>> expo init wt-app

>> cd wt-app

ทำการรันโปรแกรมโปรแกรมขึ้นมา แล้วใช้สมาร์ทโฟนรันโปรแกรม expo แล้วสั่งสแกน QR Code ที่ได้จากคำสั่ง yarn start

>> yarn start

## Source Code

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

import React from 'react';

import { StatusBar } from 'expo-status-bar';

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 ในสมาร์ทโฟน

## Passing Props

สร้างโฟลเดอร์ components ในโปรเจ็ค แล้วสร้างไฟล์ Weather.js

คอมโพเนนต์ Weather รับ Props ชื่อ zipCode

ไฟล์ Weather.js

import React from 'react';

import { Text } from 'react-native';

export default function Weather(props) {

return (

<Text>{props.zipCode}</Text>

);

}

ไฟล์ App.js

เรียกใช้คอมโพเนนต์ Weather และส่ง Props ที่ชื่อ zipCode เป็นค่า 90110

import React from 'react';

import { StatusBar } from 'expo-status-bar';

import { StyleSheet, View } from 'react-native';

import Weather from './components/Weather';

export default function App() {

return (

<View style={styles.container}>

<Weather zipCode="90110" />

<StatusBar style="auto" />

</View>

);

}

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#fff',

alignItems: 'center',

justifyContent: 'center',

},

})

## Components and Image Background

สร้างคอมโพเนนต์ Forecast ที่มีการใช้ props ดังต่อไปนี้ main, description และ temp

ไฟล์ Forecast.js

import React from 'react';

import { View, Text } from 'react-native';

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

กำหนด State โดยใช้ useState hook แล้วใช้เป็น props ส่งผ่านไปยังคอมโพเนนต์ Forecast ที่สร้างขึ้นมาใหม่

หมายเหตุ: ให้ดาวน์โหลดรูปภาพพื้นหลังมาไว้ในโฟลเดอร์นอกสุดของโปรเจค โดยตั้งชื่อไฟล์ว่า bg.jpg

import React, { useState } from 'react';

import { View, Text, ImageBackground, StyleSheet } from 'react-native';

import Forecast from './Forecast';

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%'

},

});

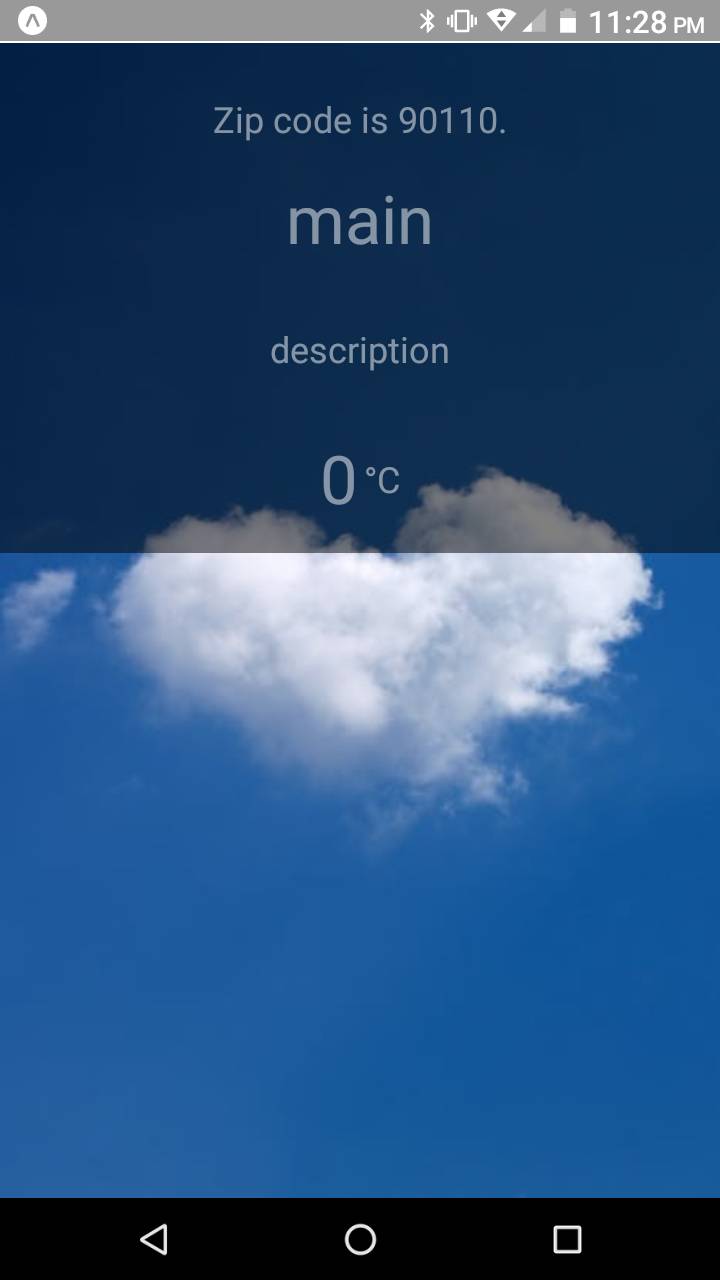
# Flex Box

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 ว่าคอมโพเนนต์ลูกควรจะจัดเรียงแนวนอน (row) และแนวตั้ง (column) โดยค่า default คือ column
* justifyContent – การกระจายตัวของคอมโพเนนต์ลูกว่าควรจะเป็นแบบแบบใด ในแนวแกนเดียวกับ flexDirection
* alignItems - – การกระจายตัวของคอมโพเนนต์ลูกว่าควรจะเป็นแบบแบบใด ในคนละแกนกับ flexDirection

ตัวอย่าง

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

CHALLENGE

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

# Connect

เพิ่ม useEffect ลงไปในคอมโพเนนต์ Weather (ทำการ sign up แบบฟรีที่ <https://home.openweathermap.org/users/sign_up> เพื่อรับ APPID)

ส่วนของไฟล์ Weather.js (ให้นำ api key ที่ได้จาก url ด้านบน มาแทนที่คำว่า YOUR\_API\_KEY)

import React, { useState, useEffect } from 'react';

import { View, Text, ImageBackground, StyleSheet } from 'react-native';

import Forecast from './Forecast';

const apiKey = 'YOUR\_API\_KEY'

export default function Weather(props) {

const [forecastInfo, setForecastInfo] = useState({

main: '-',

description: '-',

temp: 0

});

useEffect(() => {

console.log(`fetching data with zipCode = ${props.zipCode}`)

if (props.zipCode) {

fetch(`http://api.openweathermap.org/data/2.5/weather?q=${props.zipCode},th&units=metric&APPID=${apiKey}`)

.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])

...

# 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 ทั้งนี้นักศึกษาจะต้องกำหนด Style ให้เหมาะสมด้วยตนเอง

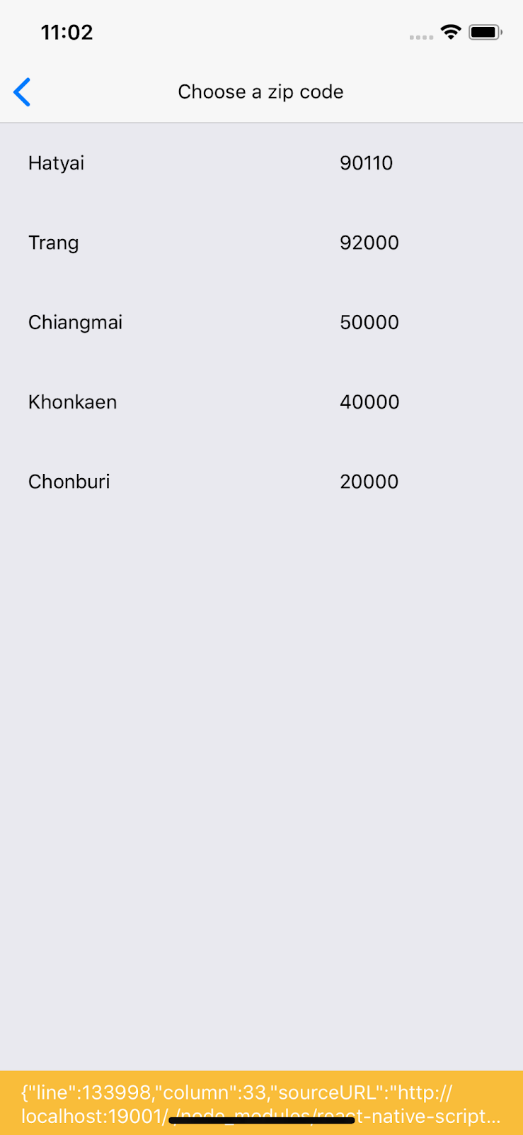
ไฟล์ ZipCodeScreen.js

import React from 'react';

import { StatusBar } from 'expo-status-bar';

import { View, Text, FlatList } from 'react-native';

import { useNavigation } from '@react-navigation/native';

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

ไฟล์ App.js

import React from 'react';

import { StyleSheet } from 'react-native';

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

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

import ZipCodeScreen from './components/ZipCodeScreen';

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 เราสามารถส่งผ่านค่าการทำงานได้โดยการใช้ Route Parameter

เพิ่มคอมโพเนนต์ WeatherScreen สำหรับหน้าจอแสดงคอมโพเนนต์ Weather

ไฟล์ Weather.js

import React from 'react';

import { View } from 'react-native';

import Weather from './Weather';

import { StatusBar } from 'expo-status-bar';

export default function WeatherScreen({route}) {

return (

<View>

<Weather zipCode={route.params.zipCode} />

<StatusBar style="auto" />

</View>

);

}

เพิ่ม Screen ลงใน App.js

ไฟล์ App.js

import React from 'react';

import { StyleSheet } from 'react-native';

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

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

import ZipCodeScreen from './components/ZipCodeScreen';

import WeatherScreen from './components/WeatherScreen';

const Stack = createStackNavigator();

export default function App() {

return (

<NavigationContainer>

<Stack.Navigator>

<Stack.Screen name="Home" component={ZipCodeScreen} />

<Stack.Screen name="Weather" component={WeatherScreen} />

</Stack.Navigator>

</NavigationContainer>

);

}

ปรับ ZipItem ในไฟล์ ZipCodeScreen.js เพื่อสร้างลิงค์ไปยังหน้า Weather พร้อมส่งผ่าน Route Parameter

ส่วนของไฟล์ไฟล์ ZipCodeScreen.js

import React from 'react';

import { StatusBar } from 'expo-status-bar';

import { View, Text, FlatList, TouchableHighlight } from 'react-native';

import { useNavigation } from '@react-navigation/native';

...

const ZipItem = ({ place, code, navigation }) => (

<TouchableHighlight onPress={() => navigation.navigate('Weather', { zipCode: code })}>

<View>

<Text>{place}</Text>

<Text>{code}</Text>

</View>

</TouchableHighlight>

)

...

# 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 ทั้งนี้ให้นศ. ทำการจับภาพหน้าจอ บันทึกไฟล์ (สามารถตั้งชื่อได้เอง) ไว้ใน root ของ project แล้ว push มาด้วย โดยหากมีการแก้ไขมากกว่าหนึ่งหน้า ก็ให้ส่งภาพมาทุกหน้า