Lab 5 : Core Components : Basic Component

1. ให้นักศึกษาทำการสร้าง New Project ใหม่ ทุกครั้งที่ทำการรันโปแกรม ในข้อ 2-8 โดยให้ Folder มีดังนี้ Mobile\<รหัสนักศึกษา>\Project ตามชื่อ Component

Expo init <path\folder\StudentID\Project name>

1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(Text Component)**

import React from "react";

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

const ViewBoxesWithColorAndText = () => {

return (

<View

style={{

flexDirection: "row",

height: 80,

padding: 40

}}

>

<View style={{ backgroundColor: "blue", flex: 0.3 }} />

<View style={{ backgroundColor: "orange", flex: 0.5 }} />

<Text>Hello World!</Text>

</View>

);

};

export default ViewBoxesWithColorAndText;

**บันทึกผลการทดลอง :**

เนื่องจากโค้ดของข้อที่ 2 นั้นไม่ได้มีการกำหนดว่าจะ Padding ด้านไหน เช่น PaddingTop ก็จะเป็นการเว้นระยะห่างจากด้านบน แต่การที่ไม่ได้กำหนดด้านที่จะ Padding นั้นจะทำการเว้นระยะห่างทั้ง 4 ด้าน (บน ล่าง ซ้าย ขวา) และหากใส่ค่า Padding เยอะเกินไปจะเป็นการเว้นระยะห่างจนมองไม่เห็นนั่นเอง (โดนทั้ง 4 ด้านบีบเข้ามาจนมองไม่เห็นกล่องสีหรือข้อความ)



ซึ่งหากมีการกำหนด Padding ตามในชีทของส่วนทฤษฎี (padding:20) จะทำให้เห็นว่า View ที่ครอบคลุม Component อื่น ๆ นั้นมีการกำหนดค่า flexDirection เป็น row ทำให้ Component ที่อยู่ภายใต้ View ตัวนี้จะอยู่ในแถวเดียวกันทั้งหมด ซึ่งก็จะมีทั้ง View ที่เป็นสีน้ำเงิน, View ที่เป็นสีแดง และ Text Component ที่เขียนว่า Hello World! ซึ่งแต่ละ Component ก็อาจจะมีขนาดไม่เท่ากัน เพราะมีการกำหนดค่า Flex ที่แตกต่างกัน



1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(Nested Text : Text Component)**

import React from 'react';

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

const BoldAndBeautiful = () => {

return (

<Text style={styles.baseText}>

I am bold

<Text style={styles.innerText}> and red</Text>

</Text>

);

};

const styles = StyleSheet.create({

baseText: {

fontWeight: 'bold'

},

innerText: {

color: 'red'

}

});

export default BoldAndBeautiful;

**บันทึกผลการทดลอง :**

ในข้อนี้จะเป็นการนำ Text Component ตัวหนึ่งไปใส่ไว้ใน Text Component อีกตัวหนึ่ง ซึ่ง Text Component ตัวที่อยู่ข้างในนั้นจะมี Style ทั้งของตัวเองและของ Text Component ตัวที่ห่อหุ้มตัวเองอยู่ด้วย เท่ากับว่า Text Component ที่อยู่ข้างในจะได้รับการกำหนด Style 2 แบบ คือทั้งตัวหนาของ baseText ที่กำหนดให้กับ Text Component ภายนอกและสีแดงของ innerText ที่กำหนดให้กับตัวเอง (โค้ดในรูปแบบนี้อาจจะใช้ในกรณีที่ต้องการกำหนด Style ที่แตกต่างกันในข้อความเดียวกัน เช่น การทำให้คำต่อไปมีความเด่นชัดมากขึ้นโดยการใส่สีตัวอักษร)

**รูปภาพผลการทดลองอยู่หน้าถัดไป**



(เนื่องจากไม่ได้ทำการปรับตำแหน่งข้อความแต่อย่างใด (ใช้ตามโจทย์) ข้อความจึงอยู่ชิดขอบซ้ายบน)

1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(Multiline : TextInput Component**)

import React, { Component } from 'react';

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

const UselessTextInput = (props) => {

return (

<TextInput

{...props} // Inherit any props passed to it; e.g., multiline, numberOfLines below

editable

maxLength={40}

/>

);

}

const UselessTextInputMultiline = () => {

const [value, onChangeText] = React.useState(‘พิมพ์ช้อความที่นี่');

// If you type something in the text box that is a color, the background will change to that

// color.

return (

<View

style={{

backgroundColor: value,

borderBottomColor: '#000000',

borderBottomWidth: 1,

}}>

<UselessTextInput

multiline

numberOfLines={4}

onChangeText={text => onChangeText(text)}

value={value}

/>

</View>

);

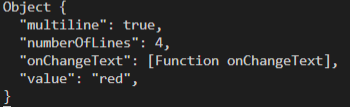
}

export default UselessTextInputMultiline;

**บันทึกผลการทดลอง :**

****

ข้อนี้จะเป็นการใช้ Props ในการส่งค่าต่าง ๆ ที่เรากำหนดในแท็ก UselessTextInput ภายในฟังก์ชันUselessTextInputMultiline ถ้าหากเราใช้คำสั่ง console.log(props) ในฟังก์ชัน UselessTextInput ก็จะได้ค่าประมาณนี้



จากรูปภาพด้านบน เป็นค่า props ที่ส่งมาในฟังก์ชัน UselessTextInput ในรูปแบบของ Object จะเห็นได้ว่าตรงกับที่เราได้กำหนดไว้ในแท็ก UselessTextInput ภายในฟังก์ชัน UselessTextInputMultiline ซึ่งจะทำให้ฟังก์ชัน Useless TextInput คืนค่าเป็น

<TextInput style={{ multiline:'true', numberOfLines:4 }}

onChangeText={text=>onChangeText(text)} value={value} editable maxLength={40}/>

หากให้อธิบายโดยรวมจะได้ความว่าโจทย์ข้อนี้เป็นการศึกษาการส่งค่า props ไปยัง TextInput Component ทำให้ TextInput มีทั้งค่า props ที่ได้รับมาและสิ่งที่เราได้กำหนดไว้เอง อย่างเช่น editable หรือ maxLength และโจทย์ข้อนี้ยังมีการกำหนดให้ว่าถ้าเราพิมพ์ข้อความที่เป็นสี เช่น red, blue นั้นจะทำให้สีพื้นหลังของ View ที่ครอบคลุม TextInput อยู่เปลี่ยนสีไปตามที่เราพิมพ์ลงไป ดังรูปภาพตัวอย่างด้านล่าง



**หมายเหตุ:**

1.multiline คือการอนุญาตให้ TextInput มีหลายบรรทัดได้หรือไม่ โดยกำหนดค่าเป็น Boolean (true คือมีหลายบรรทัดได้ false คือมีได้บรรทัดเดียว โดยค่าตั้งต้นจะเป็น false)

2.numberOfLines คือจำนวนบรรทัดของ TextInput ที่เราจะกำหนด ซึ่งต้องใช้ควบคู่ไปกับ multiline ที่กำหนดค่าเป็น true เพื่อทำให้มีได้หลายบรรทัดตามจำนวนที่กำหนดไว้

1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(ScrollView Component)**

import React from 'react';

import { StyleSheet, Text, SafeAreaView, ScrollView } from 'react-native';

import Constants from 'expo-constants';

const App = () => {

return (

<SafeAreaView style={styles.container}>

<ScrollView style={styles.scrollView}>

<Text style={styles.text}>

วิชาเหมือนสินค้า อันมีค่าอยู่เมืองไกล

ต้องยากลำบากไป จึงจะได้สินค้ามา

จงตั้งเอากายเจ้า เป็นสำเภาอันโสภา

ความเพียรเป็นโยธา แขนซ้ายขวาเป็นเสาใบ

นิ้วเป็นสายระยาง สองเท้าต่างสมอใหญ่

ปากเป็นนายงานไป อัชฌาสัยเป็นเสบียง

สติเป็นหางเสือ ถือท้ายเรือไว้ให้เที่ยง

ถือไว้อย่าให้เอียง ตัดแล่นเลี่ยงข้ามคงคา

ปัญญาเป็นกล้องแก้ว ส่องดูแถวแนวหินผา

เจ้าจงเอาหูตา เป็นล้าต้าฟังดูลม

ขี้เกียจคือปลาร้าย จะทำลายให้เรือจม

เอาใจเป็นปืนคม ยิงระดมให้จมไป

จึงจะได้สินค้ามา คือวิชาอันพิสมัยจงหมั่นมั่นหมายใจ อย่าได้คร้านการวิชา

</Text>

</ScrollView>

</SafeAreaView>

);

}

const styles = StyleSheet.create({

container: {

flex: 1,

marginTop: Constants.statusBarHeight,

},

scrollView: {

backgroundColor: 'pink',

marginHorizontal: 20,

},

text: {

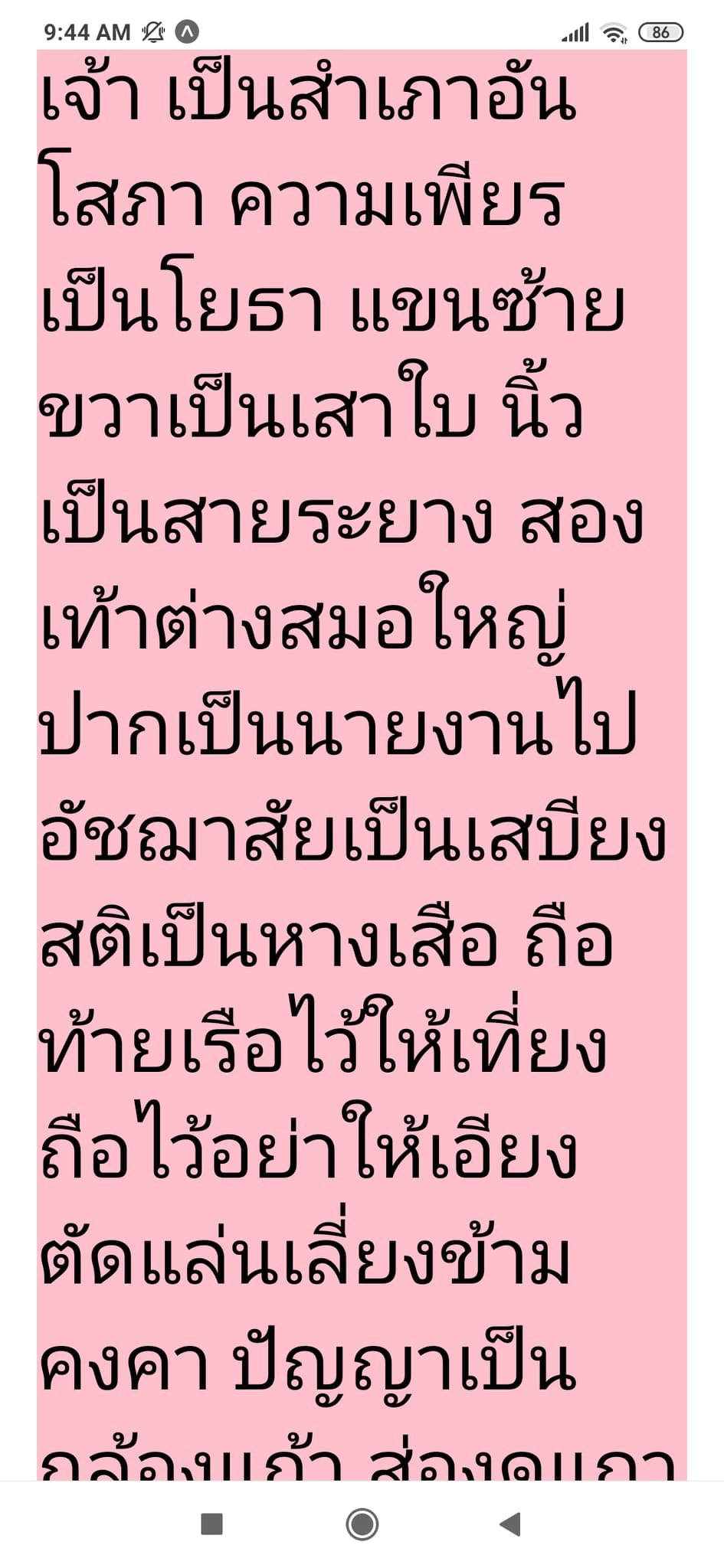
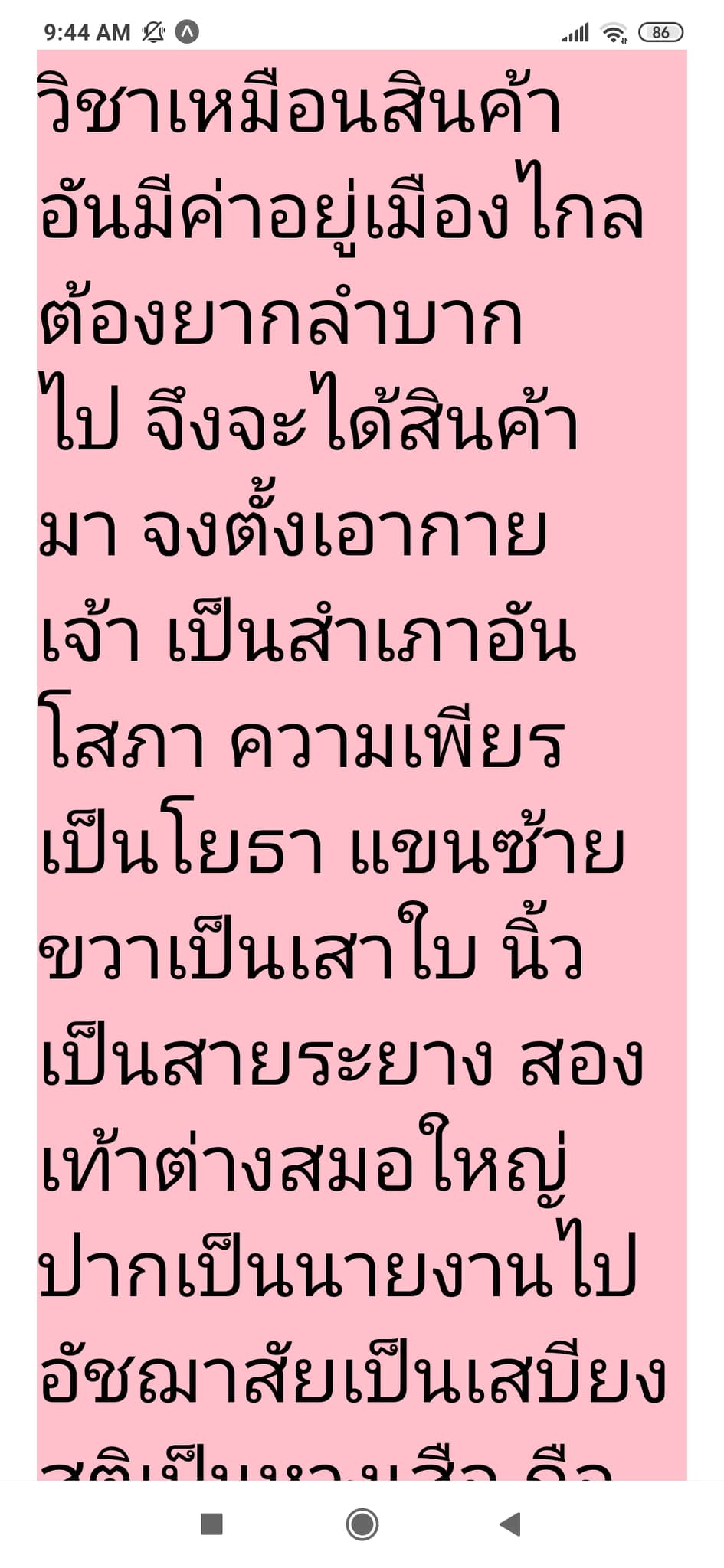
fontSize: 42,

},

});

export default App;

**บันทึกผลการทดลอง :**

****

ในข้อ 5 นี้ได้มีการใช้ SafeAreaView Component ซึ่งจะทำการแสดงผลข้อมูลภายในขอบเขตที่ปลอดภัยของหน้าจอโทรศัพท์ เช่น เว้นระยะห่างจากขอบทั้ง 4 ด้านของหน้าจอ ให้มองเห็นข้อมูลได้อย่างชัดเจน ถัดมา คือ ScrollView Component ที่จะเป็นการทำให้เราสามารถเลื่อนหน้าจอได้ในกรณีที่มีข้อความที่ยาวเกินกว่าจะแสดงผลได้หมด และสุดท้ายคือ Text Component ที่มีข้อความภายในที่ค่อนข้างยาว ทำให้เราต้องครอบ Text Component ตัวนี้ด้วย ScrollView Component เพื่อที่เราจะสามารถเลื่อนหน้าจอเพื่ออ่านข้อความที่เกินขอบเขตของหน้าจอไปได้

1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(Compose : StyleSheet View)**

import React from 'react';

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

const App = () => (

<View style={container}>

<Text style={text}>ไอที สจล</Text>

</View>

);

const page = StyleSheet.create({

container: {

flex: 1,

padding: 24,

backgroundColor: 'orange',

},

text: {

fontSize: 48,

color: '#000'

},

});

const lists = StyleSheet.create({

listContainer: {

flex: 1,

backgroundColor: '#61dafb',

},

listItem: {

fontStyle: 'italic',

fontWeight: 'bold'

},

});

const container = StyleSheet.compose(page.container, lists.listContainer);

const text = StyleSheet.compose(page.text, lists.listItem);

export default App;

**บันทึกผลการทดลอง :**

****

ในข้อ 6 จะเป็นการนำ StyleSheet มา Compose กันหรือก็คือการรวม Style เข้าด้วยกัน ในกรณีของ

const container = StyleSheet.compose(page.container, lists.listContainer);

จะเป็นการนำ container ใน page (ที่เป็น StyleSheet) มารวมกับ listContainer ใน lists (ที่เป็น StyleSheet) ซึ่งจะสังเกตได้ว่าทั้ง 2 Style นั้นมีการกำหนดสีของพื้นหลังที่แตกต่างกัน แต่ Compose นั้นจะทำการเขียนทับข้อมูล (Override) ในกรณีที่มีการกำหนด Properties เดียวกัน (กำหนด backgroundColor ทั้งคู่) โดยเอา Style ที่เราใส่ไว้ข้างหลังเป็นหลัก เช่น เรากำหนด page.container มีสีของพื้นหลังเป็นสีส้ม และกำหนด lists.listContainer ให้มีสีของพื้นหลังเป็นสีฟ้า จะทำให้ Compose เขียน container ขึ้นมาใหม่เป็น {flex:1, padding:24, backgroundColor:’’#61dafb”} ซึ่งจะเห็นได้ว่าได้ยึดตาม Properties ของ Style ตัวข้างหลังเป็นหลัก และจะรวม Properties อื่น ๆ ที่ไม่ได้กำหนดไว้ทั้งคู่ (กำหนดไว้แค่ฝั่งเดียว) เช่น padding:24 ที่กำหนดไว้ใน page.container

ในกรณีของ

const text = StyleSheet.compose(page.text, lists.listItem);

ก็มีหลักการทำงานเหมือนกันคือรวม 2 Style เข้าด้วยกันให้เป็น Style เดียว โดย Properties ที่มีการกำหนดเหมือนกัน ก็จะใช้ของ lists.listItem แทน และอันที่ไม่ได้กำหนดไว้เหมือนกันก็จะรวมเข้ามาตามปกติก็จะได้เท่ากับ

text={fontSize:48, color:’#000’, fontStyle:’italic’, fontWeight:’bold’}

แต่ในกรณีนี้จะไม่มีการกำหนด Properties เหมือนกัน ทำให้เป็นการรวม Style ตามปกติ จะเห็นได้ว่าข้อความคำว่า “ไอที สจล” นั้นมีตัวอักษรขนาด 48 เป็นสีดำ ตัวเอียง และเป็นตัวหนา

1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(absoluteFillObject : StyleSheet Component)**

import React from 'react';

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

const App = () => (

<View style={styles.container}>

<View style={styles.box1}>

<Text style={styles.text}>ก</Text>

</View>

<View style={styles.box2}>

<Text style={styles.text}>ข</Text>

</View>

<View style={styles.box3}>

<Text style={styles.text}>ค</Text>

</View>

</View>

);

const styles = StyleSheet.create({

container: {

flex: 1

},

box1: {

position: 'absolute',

top: 40,

left: 40,

width: 100,

height: 100,

backgroundColor: 'red'

},

box2: {

...StyleSheet.absoluteFill,

top: 120,

left: 50,

width: 100,

height: 100,

backgroundColor: 'blue'

},

box3: {

...StyleSheet.absoluteFillObject,

top: 120,

left: 120,

width: 100,

height: 100,

backgroundColor: 'green'

},

text: {

color: '#FFF',

fontSize: 80

}

});

export default App;

**บันทึกผลการทดลอง :**

****

ในข้อ 7 นั้นจะเป็นการทดลองใช้ Properties ได้แก่ absoluteFill และ absoluteFillObject ซึ่งทั้ง 2 ตัวนี้จะมีการกำหนดค่า Properties ต่าง ๆ ดังนี้ {position:’absolute’, top:0, left:0, right:0, bottom:0} ถ้าหากเราทดลองลบ top, left ที่กำหนดไว้ใน box2 และ box3 นั้นจะสังเกตได้ว่า View Component ที่กำหนด Style เป็น box2 และ box3 นั้นจะมี Style ที่ absoluteFill และ absoluteFillObject กำหนดให้ คือ {position:’absolute’, top:0, left:0, right:0, bottom:0} ซึ่งจะทำให้ View ทั้ง 2 ตัวนี้ซ้อนทับกันอยู่ ทำให้เราต้องกำหนด top, left ให้กับ box2 และ box3 ใหม่ (สามารถกำหนดใหม่ได้)

ซึ่งการใช้ absoluteFill และ absoluteFillObject นั้นเป็นวิธีง่าย ๆ เพื่อเป็นการกำหนดตำแหน่งของ Component ให้เป็น {position:’absolute’, top:0, left:0, right:0, bottom:0} โดย Properties ทั้ง 2 ตัวนี้สามารถใช้แทนกันและกันได้ เนื่องจากมีการกำหนดตำแหน่งต่าง ๆ เหมือนกัน โดยอาจจะใช้ในกรณีที่ต้องการความสะดวกรวดเร็ว หรือความเป็นระเบียบของโค้ด

1. ให้นักศึกษาบันทึกผลลัพธ์จากโปแกรมข้างล่างนี้ **(hairlineWidth : StyleSheet Component)**

import React from "react";

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

const App = () => (

<View style={styles.container}>

<Text style={styles.row}>คณะเทคโนโลยีสารสนเทศ</Text>

<Text style={styles.row}>สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง</Text>

</View>

);

const styles = StyleSheet.create({

container: {

flex: 1,

padding: 24

},

row: {

padding: 4,

borderBottomColor: "red",

borderBottomWidth: StyleSheet.hairlineWidth

}

});

export default App;

**บันทึกผลการทดลอง :**

****

ในข้อ 8 จะเป็นการใช้งาน hairlineWidth ให้กับทุก Component ที่กำหนด Style เป็น styles.row ซึ่งภายใน row นั้นได้มีการกำหนดให้มีเส้นขอบข้างล่างเป็นสีแดง และความหนาของเส้นขอบข้างล่างเป็น Stylesheet.hairlineWidth ซึ่งมักจะใช้เป็นความหนาชองเส้นขอบ หรือใช้ในการแบ่งระหว่าง 2 Element

โดยตัว hairlineWidth นั้นจะมีการปรับเปลี่ยนค่าให้ตรงกับพื้นฐานของแต่ละ Platform