React Native : Component

Reference: https://medium.com/thai-devmaster/%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2-android-ios-app-%E0%B8%94%E0%B9%89%E0%B8%A7%E0%B8%A2-react-native-%E0%B9%81%E0%B8%A5%E0%B8%B0-expo-%E0%B8%95%E0%B8%AD%E0%B8%99%E0%B8%97%E0%B8%B5%E0%B9%88-3-%E0%B8%84%E0%B8%AD%E0%B8%A1%E0%B9%82%E0%B8%9E%E0%B9%81%E0%B8%99%E0%B8%99%E0%B8%97%E0%B9%8C-26cec5a68bbf

React Native : Component

- เรื่องที่เป็นพื้นฐานสำคัญที่สุดของโปรแกรมที่สร้างขึ้นจาก React Native คือ คอมโพแนนท์ (Component) เนื่องจากโปรแกรมที่เราสร้างขึ้นจะเกิดจากการนำคอมโพแนนท์หลาย ๆ ตัวมาประกอบกัน ไม่ว่าจะเป็น Header, Footer, Button, Menu, checkbox, radio, picker ฯลฯ โดยที่คอมโพแนนท์แต่ละตัวจะมีหน้าที่ของตนเอง และมีการส่งข้อมูลระหว่างกันเพื่อให้ได้ผลลัพธ์ตามที่เราต้องการ
- ในหัวข้อนี้เราจะกล่างถึง
    1. คอมโพแนนท์ในโลกของ React
    2. คอมโพแนนท์แบบฟังก์ชั่น (Functional Component) และอินพุท props
    3. คอมโพแนนท์แบบคลาส (Class Component) และ state ของคอมโพแนนท์

Component ของ React

- คอมโพแนนท์ App เป็นคอมโพแนนท์ราก (Root Component) ซึ่งประกอบด้วยส่วนต่างๆดังนี้

App Component

```
JS App.js > ...
1   // import React เพื่อให้สามารถใช้ JSX ใน JavaScript ได้
2   import React from 'react';
3   import { StyleSheet, Text, View } from 'react-native';
4
5   // App component ที่สร้างขึ้นจาก JavaScript function
6   export default function App() {
7     // component จะ return JSX element
8     return (
9       <View style={styles.container}>
10        <Text>Natthapol!!!</Text>
11      </View>
12    );
13  }
14
15  // สร้าง style เพื่อใช้ในการจัดหน้าจอ หรือกำหนดรูปแบบต่างๆ
16  const styles = StyleSheet.create({
17    container: {
18      flex: 1,
19      backgroundColor: '#fff',
20      alignItems: 'center',
21      justifyContent: 'center',
22    },
23  });
24
```

Component ของ React

- บรรทัดที่ 2,3 : เรา import React และ React Native เข้ามาใช้ในคอมโพแนนท์ การ import React ทำให้เราสามารถใส่ JSX ใน JavaScript ได้
- บรรทัดที่ 6–13 : เป็นการสร้างคอมโพแนนท์ App ที่สร้างขึ้นจากฟังก์ชั่น JavaScript และ return คอมโพแนนท์ View และ Text ที่เขียนอยู่ในรูป JSX
- บรรทัดที่ 16–23 : สร้าง styles อ็อบเจกต์ ซึ่งเป็น JavaScript อ็อบเจกต์ ที่ใช้ในการกำหนด style ให้กับคอมโพแนนท์
- คอมโพแนนท์สามารถรับอินพุทผ่านทาง props และ return เป็น JSX element
- เราสามารถสร้างคอมโพแนนท์ได้ 2 แบบ คือ คอมโพแนนท์แบบฟังก์ชั่น (Functional Component) และคอมโพแนนท์แบบคลาส (Class Component)

Component แบบฟังก์ชั่น (Functional Component) และอินพุท props

- คอมโพแนนท์แบบฟังก์ชั่น คือ คอมโพแนนท์ที่สร้างขึ้นจากฟังก์ชั่น JavaScript โดยรับ **props** เป็นพารามิเตอร์ และ **return** ค่าออกมาเป็น **JSX element** ดังตัวอย่าง
- ตัวอย่างนี้เป็นฟังก์ชั่นที่ทำหน้าที่รับ **text** เข้ามาและแสดงออกทางหน้าจอ

components > JS MyTextOutput.js > ...

```
1   import React from 'react';
2   import { View, Text} from 'react-native';
3
4   // MyTextOutput เป็น functional component (เขียนแบบ arrrow function)
5   const MyTextOutput = (props) => {
6       // แสดง text ที่รับเป็น input เข้ามา
7       return (
8           <View>
9               <Text>{props.text}</Text>
10          </View>
11      );
12  }
13
14  export default MyTextOutput;
```

คอมโพแนนท์ **MyTextOutput** เป็นคอมโพแนนท์แบบฟังก์ชั่น

Component แบบฟังก์ชั่น (Functional Component) และอินพุท props

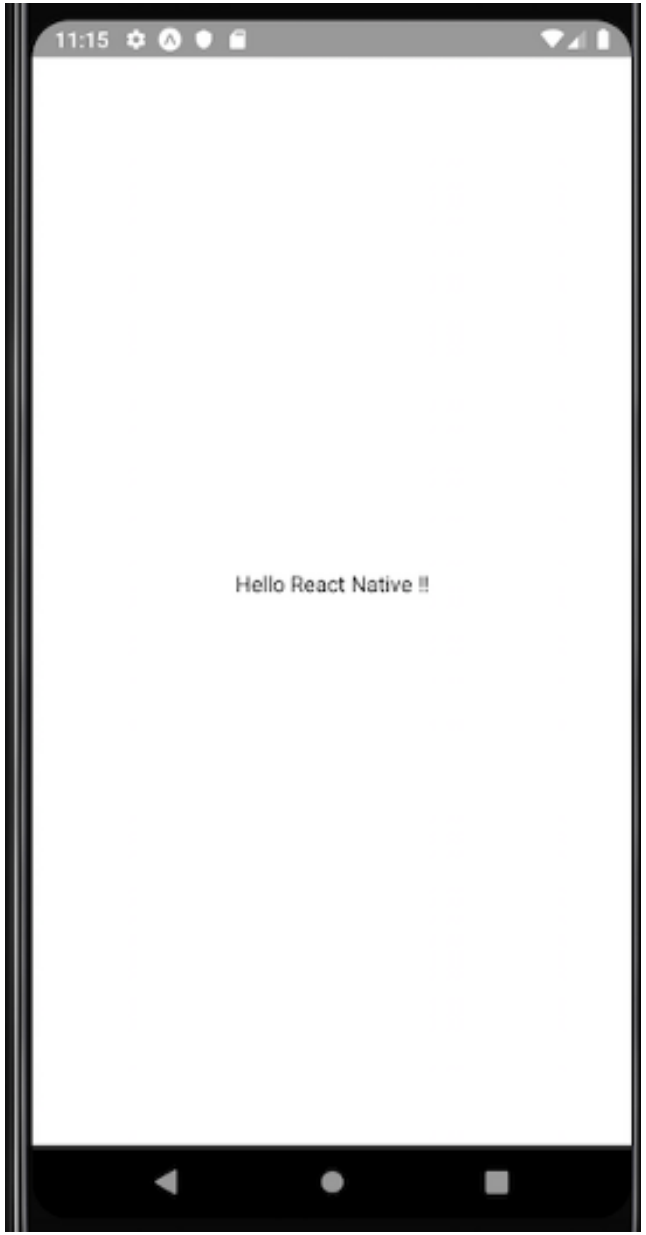
- เราสามารถนำคอมโพแนนท์ไปใช้งานโดยการ import ดังตัวอย่างผมนำคอมโพแนนท์ MyTextOutput มาใช้ในคอมโพแนนท์ App
- **บรรทัดที่** 5 : import คอมโพแนนท์ MyTextOutput มาใช้งานในคอมโพแนนท์ App
- **บรรทัดที่** 13 : เราสามารถใช้คอมโพแนนท์ในรูปของ JSX ได้ โดยสามารถส่งอินพุทผ่านทางพร็อพเพอตี้ที่ชื่อว่า text

App.js > App

```
1   // import React เพื่อให้สามารถใช้ JSX ใน JavaScript ได้
2   import React from 'react';
3   import { StyleSheet, Text, View } from 'react-native';
4
5   // import คอมโพแนนท์ MyTextInput มาใช้งาน
6   import MyTextOutput from './components/MyTextOutput';
7
8   // App component ที่สร้างขึ้นจาก JavaScript function
9   export default function App() {
10    // component จะ return JSX element
11    return (
12      <View style={styles.container}>
13        <MyTextOutput text="Hello React Native !!"></MyTextOutput>
14      </View>
15    );
16  }
17
18  // สร้าง style เพื่อใช้ในการจัดหน้าจอ หรือกำหนดรูปแบบต่างๆ
19  const styles = StyleSheet.create({
20    container: {
21      flex: 1,
22      backgroundColor: '#fff',
23      alignItems: 'center',
24      justifyContent: 'center',
25    },
26  });
```

คอมโพแนนท์ App

- เมื่อรันจะได้ผลลัพธ์ดังนี้



Component แบบคลาสและ state ของ Component

- นอกจากคอมโพแนนท์แบบฟังก์ชั่นแล้วใน React เราสามารถเขียนคอมโพแนนท์แบบคลาสได้ด้วยดังตัวอย่าง
- คอมโพเนนท์ MyTextOutput ที่เขียนแบบคลาสนี้ทำงานเหมือนกับเวอร์ชั่นที่เขียนด้วยฟังก์ชั่นทุกประการ
- **บรรทัดที่** 5 : เขียนคอมโพแนนท์ MyTextOutput ใหม่โดยประกาศเป็นคลาส ซึ่งสืบทอดมาจาก React.Component
- **บรรทัดที่** 8 : เราต้องสร้างเมธอท render โดยเมธอทนี้จะ return JSX อิลิเมนท์ออกมาเพื่อแสดงที่หน้าจอ
- **บรรทัดที่** 11 : เราจะใช้งาน props เป็นอินพุทผ่าน this เนื่องจากเวลาคอมโพแนนท์ถูกเรียกใช้งานมันจะถูกสร้างขึ้นมาเป็นอ็อบเจกต์

components > JS MyTextOutput.js > ...

```
1   import React from 'react';
2   import { View, Text } from 'react-native';
3
4   // MyTextOutput เป็นคอมโพแนนท์แบบคลาส
5   class MyTextOutput extends React.Component {
6       // แสดง text ที่รับเป็น input เข้ามา
7       // การใช้งาน props ต้องเรียกผ่าน this
8       render() {
9           return (
10              <View>
11                  <Text>{this.props.text}</Text>
12              </View>
13          );
14      }
15  }
16
17  export default MyTextOutput;
```

คอมโพแนนท์แบบคลาส (Class Component)

Component แบบคลาสและ state ของ Component

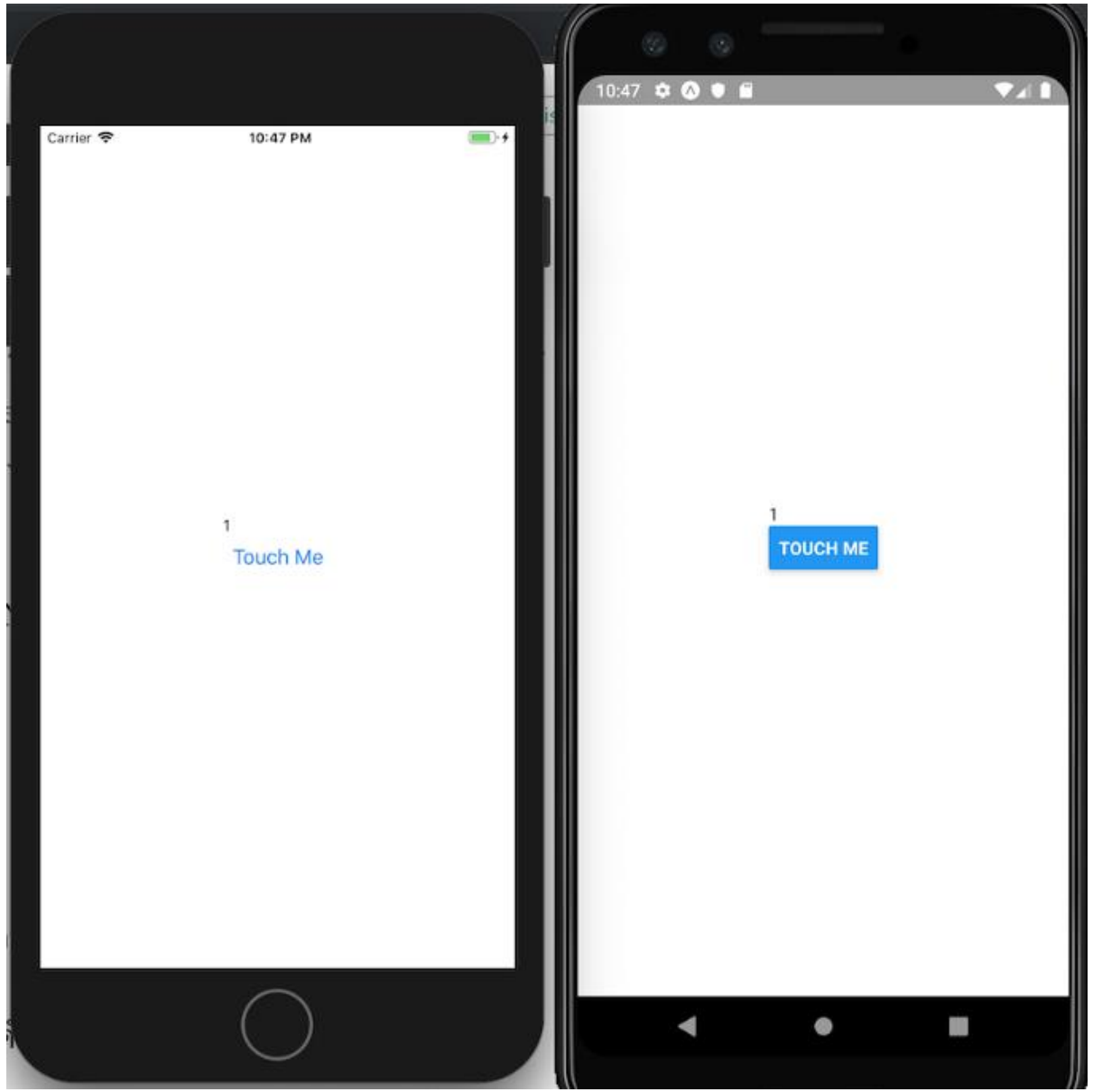
- คอมโพแนนท์แบบคลาสนอกจากจะมีความสามารถรับอินพุทผ่านทาง **props** แล้วมันยังสามารถเก็บ **state** ได้ด้วย ดังตัวอย่าง

components > JS MyTouchCounter.js > ...

```
1  import React from 'react';
2  import { View, Text, Button } from 'react-native';
3
4  class MyTouchCounter extends React.Component {
5      // กำหนดค่าเริ่มต้นให้ counter = 0
6      state = {
7          counter: 0
8      }
9
10     // เมธอดสำหรับจัดการอีเวนท์เมื่อมีการ touch ปุ่ม
11     handleTouchMe = () => {
12         // อัพเดท counter เพิ่ม 1 ทุกครั้งเมื่อเกิดอีเวนท์ touch
13         this.setState({
14             counter: this.state.counter + 1
15         });
16     }
17
18     render() {
19         return (
20             <View>
21                 <Text>{this.state.counter}</Text>
22                 <Button title="Touch Me" onPress={this.handleTouchMe}></Button>
23             </View>
24         );
25     }
26 }
27
28 export default MyTouchCounter;
29
```

- **บรรทัดที่** 7 : กำหนดค่าเริ่มต้นของ counter = 0 อยู่ภายใต้อ็อบเจกต์ state
- **บรรทัดที่** 11 : สร้าง callback เมธอทสำหรับจัดการอีเวนท์กดปุ่ม เราส่งเป็นอินพุทให้ Button ผ่านทางพร็อพเพอร์ตี้ onPress
- **บรรทัดที่** 15 : เพิ่มค่าของ counter ทีละ 1 เมื่อมีการกดปุ่ม
- การอัพเดทค่าใน state ต้องทำผ่านเมธอท setState() เท่านั้นห้ามกำหนดค่าให้กับ state ตรงๆ เช่น this.state.counter = 1
- เนื่องจาก React มีกระบวนการภายในที่ใช้ตรวจสอบว่ามีการเปลี่ยนแปลงค่าของ state และจะเรียกเมธอท render() เพื่อทำการเปลี่ยนแปลงหน้าจอ
- กระบวนการนี้เรียกว่าวงจรชีวิตของคอมโพแนนท์ (Component Lifecycle)
- **บรรทัดที่** 21 : คอมโพแนนท์ Text ของ ReactNative มีหน้าที่แสดงข้อความ
- การเขียน <Text>{this.state.counter}</Text> จะเห็นว่าเราเขียนคำสั่งอยู่ภายใต้ {} เพื่อบอกว่าค่าในนี้เป็นคำสั่ง JavaScript (JavaScript Expression) ซึ่งในที่นี้มีความหมายว่าให้อ่านค่า counter ออกมาให้นำไปแสดงในคอมโพแนนท์ Text
- **บรรทัดที่** 22 : คอมโพเนนท์ปุ่ม ในตัวอย่างนี้ส่งพร็อพเพอร์ตี้ ไป 2 ตัวคือ title และ onPress
  - title รับข้อความเพื่อนำไปแสดงที่ปุ่ม
  - onPress รับฟังชั่น callback ที่ทำงานเมื่อมีการกดปุ่ม

- ลองรันโปรแกรมได้ผลดังรูป
- เมื่อเรากดปุ่ม counter จะเพิ่มขึ้นทีละ 1



สรุป

- คอมโพแนนท์ของ React สามารถเขียนได้ 2 แบบ คือ คอมโพแนนท์แบบฟังก์ชั่น(Functional Component) และคอมโพแนนท์แบบคลาส(Class Component)
- ทั้งสองแบบรับค่า props เป็นอินพุทเหมือนกัน
- คอมโพแนนท์แบบคลาส สามารถเก็บอ็อบเจกต์ state ได้ ซึ่งการเปลี่ยนค่า state ต้องทำผ่านเมธอท setState() เท่านั้นห้ามกำหนดค่าตรงๆ
- เราสามารถจัดการอีเว็นท์ ที่เกิดขึ้นจากคอมโพแนนท์ด้วยการส่งฟังก์ชั่น callback ผ่านทาง props ดังตัวอย่างอีเวนท์ onPress ของปุ่มกด

**หมายเหตุ ตั้งแต่ React เวอร์ชั่น 16.8 เป็นต้นไป ทีมพัฒนาได้ออกฟีเจอร์ใหม่ชื่อว่า React Hooks ซึ่งส่งผลให้คอมโพแนนท์แบบฟังก์ชั่นสามารถเก็บ state ได้แล้ว